UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

MARCOS ANTONIO ESTREMOTE

Gerenciamento de memória através da utilização de tabelas de dispersão em um módulo híbrido com suporte ao protocolo CAN (*Controller Area NetWork*) e ao padrão 802.15.4 *ZigBee*.

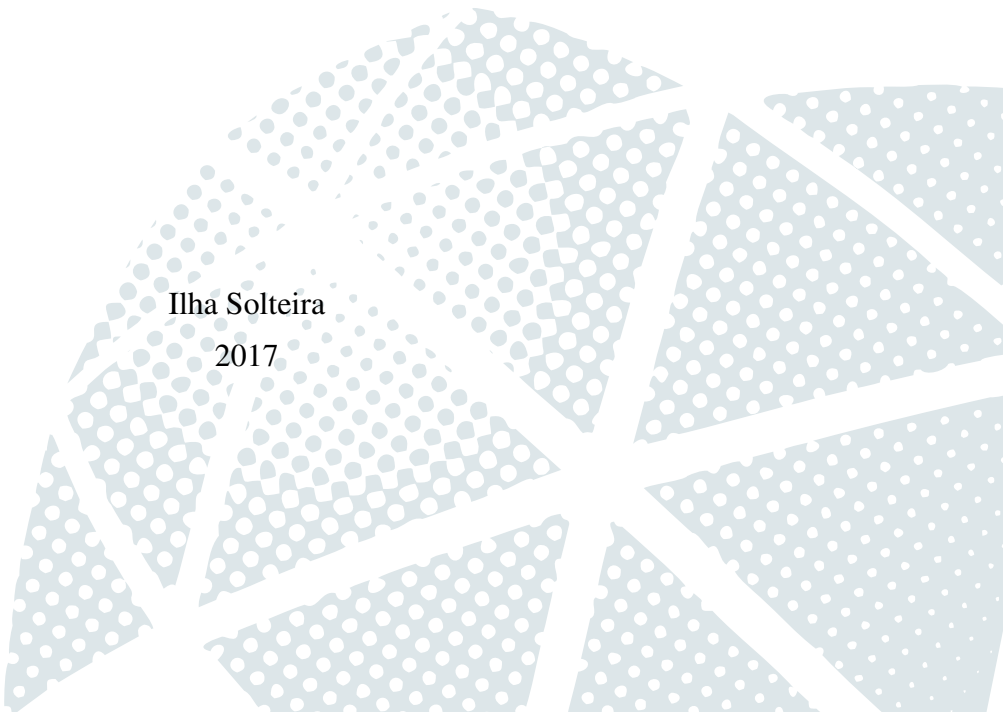
Ilha Solteira
2017

MARCOS ANTONIO ESTREMOTE

Gerenciamento de memória através da utilização de tabelas de dispersão em um módulo híbrido com suporte ao protocolo CAN (*Controller Area NetWork*) e ao padrão 802.15.4 *ZigBee*.

Tese apresentada à Faculdade de Engenharia do Câmpus de Ilha Solteira - UNESP como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.
Área do conhecimento : Automação.

Prof. Dr. Nobuo Oki
Orientador



Ilha Solteira
2017

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

E823g

Estremote, Marcos Antonio.
Gerenciamento de memória através da utilização de tabelas de dispersão em um módulo híbrido com suporte ao protocolo CAN(Controller Area Network) e ao padrão 802.15.4 “ZigBee” / Marcos Antonio Estremote. -- Ilha Solteira: [s.n.], 2017
129 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia. Área de conhecimento: Automação, 2017

Orientador: Nobuo Oki
Inclui bibliografia

1. Padrão Ieee 802.15.4. 2. Protocolo Can (Controller Area Network). 3. Tabelas de dispersão (Hash Tables). 4. Protocolos heterogêneos. 5. Protocolo híbrido. 6. Redes de sensores sem fio.

CRB8- 4740

unesp

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: Gerenciamento de memória através da utilização de tabelas de dispersão em um módulo híbrido com suporte ao protocolo CAN (Controller Area Network) e ao Padrão 802.15.4

AUTOR: MARCOS ANTONIO ESTREMOTE
ORIENTADOR: NOBUO OKI

Aprovado como parte das exigências para obtenção do Título de Doutor em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. NOBUO OKI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. SUELY CUNHA AMARO MANTOVANI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. DALVA MARIA DE O VILLARREAL
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. EDSON ANTONIO BATISTA
Faculdade de Engenharias, Arquitetura e Urbanismo / Universidade Federal de Mato Grosso do Sul

Prof. Dr. MÁRIO ANDERSON DE OLIVEIRA
Departamento de Eletro-eletrônica / Instituto Federal de Mato Grosso

Ilha Solteira, 24 de fevereiro de 2017

À minha família, sobretudo meus pais:
Lenir Almeida Estremote e
João Antonio Estremote “In Memoria”,
pela educação pautada, pela honestidade
e trabalho, meu profundo reconhecimento e consideração.

Às minhas filhas Izabella Prato Estremote,
Maria Eduarda Prato Estremote
e minha esposa
Elaine Prato Estremote,
pontos de luz, que a cada dia me fortalece e edifica para a
conquista dos meus sonhos.

Ofereço!

“Porque ficar de braços cruzados se o maior homem do mundo morreu de braços abertos,
pois vitorioso não é aquele que vence o outro, mas sim a si mesmo”.

AGRADECIMENTOS

A minha gratidão, primeira e principalmente, é a Deus, forte rochedo em que me abrigo autor e guia de nossas vidas, por tanto amor e tanto cuidado a mim dispensados.

Agradeço a meus pais, minha esposa (Elaine Prato Estremote), minhas filhas, aos meus irmãos (Mário Márcio Estremote e Marcelo Estremote) e familiares, que sempre lutaram por seus ideais me ensinando, com isso, a persistir na caminhada, por mais árdua que ela seja.

Ao professor Doutor Nobuo Oki, que com sua competência e habilidade, unido à sua receptividade, amizade e confiança se fez presente em toda esta caminhada.

Meus agradecimentos a todos os professores e funcionários da FEIS-UNESP, que direta ou indiretamente contribuíram para a realização deste trabalho.

Aos amigos Tiago da Silva Almeida, Márcio Geraldo Cidin Bonzegno, Tércio Alberto dos Santos Filho, Maicon Sartin, Marcus Vinícius Alves Pereira, Flavilene Souza, Rodrigo Rinaldi Barretos, Ródney dos Santos Mazuqui, José Paulo Codinho, Virgílio Fries Muller, Valdo Fernando Costa, Marcos Aurélio Suman, Cleber Rodrigues da Silva e Ricardo Rodrigues da Silva que os considero um exemplo a ser seguido tanto no campo profissional como pessoal.

A todos os colegas e amigos que direta ou indiretamente, cooperaram para a realização deste trabalho.

“A persistência é o menor

caminho do êxito”. (Charles Chaplin)

RESUMO

A utilização de redes de comunicação sem fio deixou de ser uma ferramenta opcional para tornar-se uma necessidade no monitoramento de residências, automóveis, controles de processos automatizados e comunicação entre as pessoas. Tratando-se de redes de transdutores cabeadas, as redes CAN (*Controller Area Network*) são utilizadas em automóveis modernos, instrumentação médica, em veículos táticos, na automação de processos, no transporte metropolitano e em sistemas de controle de fábricas. A maioria das estruturas críticas de sistemas de controle fazem uso do CAN em algum ponto na rede, para conectar sensores que se encontram distantes e controlar atuadores de um sistema, ou para conectar vários controladores que utilizam uma interface em comum. O padrão sem fio, IEEE 802.15.4, comercialmente conhecido como "ZigBee", foi projetado para operar em baixas taxas de dados, com segurança e facilidade de configurações de rede. Esta tese tem como objetivo desenvolver um sistema heterogêneo utilizando microcontroladores ATMEGA em que, o modelo do protocolo CAN e o padrão IEEE 802.15.4 estejam acoplados. Este módulo será capaz de gerenciar e monitorar sensores e atuadores utilizando CAN e, através do padrão sem fio 802.15.4, comunicar-se com os outros módulos da rede. O interfaceamento entre os pacotes da rede de controle de área (CAN) com a rede ZigBee é realizado através da implementação de tabelas de dispersão (*Hash Tables*) para o gerenciamento e otimização da memória utilizada. As análises realizadas de tempo de inserção, remoção e comunicação de dados, com o auxílio da técnica computacional de tabelas de dispersão para o armazenamento das informações, mostram que este procedimento favorece a comunicação entre os protocolos de redes industriais com os protocolos de redes sem fios. A economia de memória do microcontrolador com a utilização das tabelas de dispersão proposta nesta tese chegou a ser em média 750% superior do que as que não se utilizam das tabelas de dispersão.

Palavras-chave: Protocolo CAN. IEEE 802.15.4. Redes de sensores sem fio. ATMEGA. AVR studio. Arduino. Tabelas de dispersão.

ABSTRACT

The use of wireless communication networks is not an optional tool and become a requirement in automated systems, such as monitoring home, automobiles, automated process control and communication between people. In another aspect, wired networks, such as CAN networks, are used in modern automobiles, medical instrumentation, tactical vehicles, process automation, metropolitan transport and manufactory control systems. Many critical structures in control system use CAN network at some point, to connect sensors that are far away and to control system actuators, or to connect several controllers that use a common interface. The wireless IEEE standard 802.15.4, commercially known as "ZigBee", is designed to operate at low data transfer rates, with security and facility of network configurations. This thesis aims to develop a heterogeneous system using ATMEGA microcontrollers in which the CAN protocol model and the IEEE standard 802.15.4 are coupled. This module is capable of managing and monitoring sensors and actuators using CAN and, through the IEEE standard 802.15.4, communicating with the other modules in network. The interface between the CAN network packets with the ZigBee network is performed through the implementation of Hash Tables to manage and optimize the memory used. The analysis of time of insertion, delete and data communication, with the aid of the computational technique of hash tables for the storage of information, show that this procedure favors the communication between the protocols of industrial networks with protocols of wireless networks . The memory economy of the microcontroller with the the hash tables proposed in this thesis was on average 750% higher than those without the hash tables.

Keywords: CAN protocol. IEEE 802.15.4. Wireless sensor networks. ATMEGA. AVR studio. Arduino. Hash tables.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ACK	*Acknowledgement*
ACL	*Access Control List*
AGV	*Autonomous Guided Vehicles*
AMS	*Asset Management System*
APC	*Advanced Process Control*
APP	Aplicação
APS	*Advanced Planning and Scheduling*
ASI	*Automation Systems Interconnect*
BRAM	*Block Random Access Memory*
CAN	*Controller Area Network*
CiA	*CAN in Automation*
CLP	Controladores Lógicos Programáveis
CMI	*Code Message Integrity*
CRC	*Cyclic Redundancy Code*
CSMA/CD+AMP	*Carrier Sense Multiple Access/Collision Detection and Arbitration on Message Pr*
CSMA/CR	*Carrier Sense Multiple Access/Collision Resolution*
CSMA-CA	*Carrier Sense Multiple Access - Collision Avoidance*
dB	Decibéis
DC	*Direct Current*
DDR	*Double Data Rate*
DLC	*Data Length Code*
DLL	*Data Link Layer*
DP	*Decentralized Peripherals*
DSSS	*Direct Sequence Spread Spectrum*
DW	*Data Warehousing*
E/S	Entrada e Saída
ECU	Unidade de Controle Eletrônica
EIS	*Executive Information Systems*
EOF	*End of File*
ERP	Enterprise Resource Planning
FEIS	Faculdade de Engenharia de Ilha Solteira

FMS	*Fieldbus Message Specification*
FFD	*Full Function Device*
FPGA	*Field Programmable Gate Array*
Gbps	Giga bits por segundos
GTS	*Guaranteed Time Slot*
HVAC	*Heating, Ventilation and Air Conditioning*
I/O	*Input/Output*
IDE	*Integrated Development Environment*
IDE	*Identifier Extension*
IEEE	*Institute of Electrical and Electronic Engineers*
IHM	Interface Homem Máquina
IP	*Internet Protocol*
ISA	*Industry Standard Architeture*
ISO	*International Organization for Standardization*
IVN	*IntraVehicle Network*
KB	Kilo Bytes
Km	Kilômetros
LIMS	*Lab Information System*
LLC	*Logical Link Control*
LR-WPAN	*Low Rate - Wireless Personal Area Network*
m	Metros
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
Mbits/s	Mega bits por segundos
MES	*Manufacturing Execution System*
MFR	*MAC Footer*
MHR	*MAC Header*
MHz	Mega Hertz
MMS	*Maintenance Management System*
MPDU	*MAC Control Data Unit*
MSDU	*MAC Service Data Unit*
NWK	Camada de Rede
OSI	*Open Systems Interconnection*
PAN	*Personal Area Network*
PCI	*Peripheral Component Interconnect*

PHY	Camada Física
PIMS	*Process Information Management System*
PPDU	*PHY Service Data Unit*
RAM	*Random Access Memory*
RF	Rádio Frequência
RFD	*Reduce Function Device*
RS-232	*Recommend Standard-232*
RSSF	Redes de Sensores sem Fio
RTR	*Remote Transmission Request*
SCADA	*Supervisory Control and Data Aquisition*
SDCD	Sistemas Digitais Remotos de Controle Distribuído
SIG	Sistemas de Informações Gerenciais
SOF	*Start of Frame*
SPI	*Serial Peripheral Interface*
SRR	*Substitute Remote Request*
TA	Tecnologia de Automação
TI	Tecnologia da Informação
TQFP	*Thin Quad Flat Package*
UNESP	Universidade Estadual Paulista
USB	*Universal Serial Bus*
V	Volts
WAN	*Wide Area Network*
WPAN	*Wireless Personal Area Network*
ZIF	*Zero Insertion Force*

SUMÁRIO

1 INTRODUÇÃO GERAL

Atualmente o gerenciamento de memória é essencial em dispositivos que possuem limitação de expansão ou que possuam o aumento de adição de memória como fator de inviabilidade. Técnicas de computação e estruturas de dados em alto nível fornecem subsídios para o gerenciamento de memória em dispositivos onde a memória é limitada, como nos microcontroladores. Memória é um componente fundamental para que programas possam ser executados corretamente em dispositivos embarcados.

A integração dos protocolos de comunicação de redes industriais cabeadas com os de redes sem fio se mostram importantes e essenciais na comunicação de ambientes industriais. Assim, o estudo de técnicas para a confecção de tecnologias computacionais para melhorar e diminuir gastos com implementação física e de infraestrutura é primordial. O cenário de integração de um protocolo que se utilize da comunicação sem fio com um protocolo de redes industriais em um único *chip* é desejável. Sendo assim, é importante o estudo desta integração e implementação em dispositivos microcontrolados que possuam limitação na expansão de memória.

Com o problema definido, foram abordadas as principais redes industriais existentes e as principais tecnologias de comunicação sem fios implantadas. Além, de trabalhos que deram suporte ao desenvolvimento do projeto.

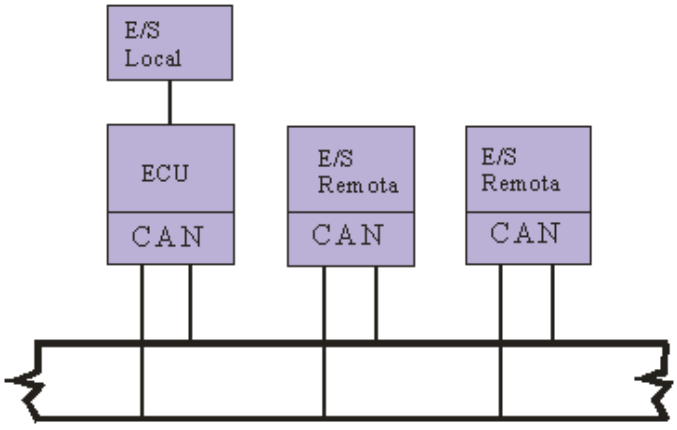
No trabalho de Kuban (2006), foi realizada a integração dos protocolos CAN com o padrão sem fio IEEE 802.15.4, no qual este trabalho foi baseado para a integração dos protocolos CAN e IEEE 802.15.4, e também foram abordadas outras possibilidades de utilização de outras tecnologias e padrões de redes para o problema. Kuban (2006) realizou a comunicação dos protocolos em microcontroladores diferentes, para isso, houve a necessidade de configurar cada microcontrolador da rede com seu protocolo específico. A técnica de Kuban (2006) acaba por aumentar o valor final do desenvolvimento da aplicação. Nesta tese é realizada uma proposta de integração em um mesmo dispositivo microcontrolador dos dois protocolos (CAN e IEEE 802.15.4), criando assim, uma plataforma híbrida para o controle destas tecnologias.

Desenvolvido por Robert Bosch, em 1991, a Rede de Controle de Área - *Controller Area Network* (CAN) (BOSCH, 1991) é utilizada em automóveis modernos, instrumentação médica, em veículos táticos, na automação de processos, no transporte metropolitano e em sistemas de

controle de fábricas. O *DeviceNet*[1] é um exemplo de uma rede comercial SCADA (*Supervisory Control and Data Aquisition* - Sistemas de Supervisão e Aquisição de Dados), que tem como base a especificação CAN.

Este sistema é utilizado para realizar a conexão entre nós sensores com um baixo custo através de uma rede. A maioria das estruturas críticas de sistemas de controle fazem uso do CAN, em algum ponto na rede, para conectar sensores remotos e controlar atuadores de um sistema, ou para conectar vários controladores que utilizam uma interface em comum. Em uma rede CAN, cada dispositivo pode enviar e receber mensagens, somente se o barramento de comunicação estiver livre. O aspecto mais importante do protocolo CAN é que a transmissão (codificação) das mensagens é realizada por conteúdo da mensagem e não por endereço, como na maioria dos protocolos. Cada nó CAN determina se a mensagem é significativa, em seguida, um filtro pré-determinado combina com os dados no campo do identificador do pacote recebido e determina se a mensagem é ou não de prioridade para tomar posse do barramento. Este mecanismo permite a comunicação por *broadcast* ou *multicast* de mensagens, em que um nó pode enviar a mesma mensagem para muitos nós, um aspecto potencialmente útil na arquitetura de processamento distribuído "mestre-escravo" (HENNESSY; PATTERSON, 2011). O sincronismo dos bits, a taxa de dados, e o comprimento da mensagem são variáveis programadas com base na capacidade física da conexão de rede. Na Figura 1 apresenta-se um modelo de rede baseado no barramento CAN com uma Unidade de Controle Eletrônica (ECU).

Figura 1 – Configuração típica do Barramento CAN



Fonte: Próprio Autor

[1] DeviceNet é um protocolo de comunicação utilizado em automação industrial para troca de dados entre dispositivos de controle. Ela utiliza *Controller Area Network* e define uma camada de aplicação para cobrir uma faixa dos perfis de dispositivos. Aplicações típicas incluem troca de comunicação, dispositivos de segurança e controle de entrada/saída de redes.

A especificação CAN, publicada pela Bosch, normalmente realiza os serviços associados à camada de ligação de dados do modelo OSI[2]. Várias pilhas de protocolos[3] comerciais disponíveis fornecem serviços de nível superior, mas esses produtos normalmente requerem um investimento substancial do hardware proprietário. A rede CAN opera com baixas taxas de dados, geralmente, ao longo de um simples par trançado, mas muitos mecanismos de conexão são válidos de acordo com o padrão da camada física, que é especificado na norma ISO (*International Organization for Standardization*) 11898.

Ao contrário dos diferentes esquemas de endereços (origem/destino) encontrados em protocolos como o IP (*Internet Protocol*), uma característica única no protocolo CAN é a utilização de identificadores de filtros de mensagem.

A maior área de utilização do protocolo CAN continua sendo nos modernos sistemas eletrônicos, sendo que, vários automóveis utilizam os modelos do CAN que estão atualmente disponíveis. Nos automóveis, o sistema CAN é aplicado em redes IntraVeiculares (IVN - *IntraVehicle Network*) e, também, em outras áreas, como por exemplo, no controle de motores e distribuição de áudio. Outras aplicações automotivas incluem aquecimento, controle de iluminação e sistemas de informação e entretenimento. Além do padrão para automóveis, a rede CAN é utilizada em caminhões, comunicação de caminhões reboque; em trens, para controle de portas, controle de freio e validação de dispositivos através de cartões; na eletrônica marítima, controle de bombas e válvulas, nos aviões, para sensores de vôo e sistemas de navegação; em equipamentos médicos, para gestão de equipamentos em salas de operação, em sistemas de automação de fábricas, além de processos de controle e aquisição de dados remotos.

Importantes implementações da rede CAN tem sido em produtos com tecnologia sem fio (*wireless*). Esses dispositivos sem fios são desejáveis nas situações em que uma conexão física pode ser de alto custo, perigosa ou para controle de veículos autônomos (KONGEZOS; ALLEN, 2002).

A falta de preocupação com a segurança dos sistemas computacionais os torna vulneráveis a ataques ou a invasões. O bloqueio de um sinal sem fio da rede CAN pode impedir a transmissão de dados em tempo real, proporcionando resultados catastróficos. Além disso, se o CAN RF é acessível, *hackers* podem causar uma falha do sistema enviando dados impróprios na rede (BYRES, 1999) quando o estado do barramento estiver recessivo. Neste trabalho, a

[2] *Open Systems Interconnection*, ou Interconexão de Sistemas Abertos, é um conjunto de padrões ISO relativo à comunicação de dados.

[3] Conjunto de regras que regem o formato e significado dos pacotes ou mensagens que são trocadas por uma camada e sua entidade par em outra máquina. A camada usa protocolos para implementar suas definições de serviço e está livre para alterá-los, desde que não mude o serviço visível para os seus usuários (TANENBAUM, 2003b).

rede não oferecerá nenhum mecanismo para impedir ouvintes desautorizados, possivelmente concorrentes de adquirir dados da rede em questão.

O padrão sem fio, IEEE 802.15.4, foi concluído no final de 2003. Comercialmente conhecido como "ZigBee", este sistema é projetado para operar em baixas taxas de dados, com segurança e facilidade nas configurações de rede. Esta rede é comumente conhecida LR-WPAN (*Low-Rate Wireless Personal Area Network*). LR-WPANs são utilizados em redes domésticas, instrumentação médica e outras aplicações que exigem sensores remotos de baixa potência, a fim de aumentar o tempo de vida da bateria e minimizar a manutenção do sensor. Dois elementos chaves do padrão IEEE 802.15.4 LR-WPAN podem ser destacados: operação com baixa potência e a implementação inerente da segurança.

O padrão IEEE 802.15.4 especifica as pilhas de protocolo de Camada de Acesso ao Meio (*Medium Access Control* - MAC) e a Camada Física (PHY). Simplesmente, o RF (Rádio Frequência) na camada física é análogo entre dois nós na comunicação. A camada física do LR-WPAN usa o espectro de propagação da sequência direta (*Direct Sequence Spread Spectrum* - DSSS), que oferece resistência inerente de bloqueio. A subcamada MAC define a estrutura do empacotamento da mensagem e o estabelecimento de uma conexão. No padrão IEEE 802.15.4 o mecanismo utilizado para controle do barramento é o CSMA-CA[4] (*Carrier Sense Multiple Access - Collision Avoidance*). Os recursos de segurança também são executados e incluem a capacidade de manter uma lista de Controle de Acesso (*Access Control List* - ACL) e a capacidade de executar criptografia simétrica (COMMITTEE et al., 2003).

Uma conexão sem fio para o acesso CAN é utilizada em redes IVN e SCADA, principalmente para o acesso às partes rotativas (sensor de pressão e incêndio), aplicações de sensoriamento remoto e Veículos Autônomos Guiados (*Autonomous Guided Vehicles* - AGV). Poucas soluções de interfaces comerciais sem fio CAN estão disponíveis e há poucas características de compatibilidade entre estes projetos proprietários. As implementações atuais são baseadas no padrão IEEE 802.11 ou *Bluetooth*, ambas são aceitáveis, mas, a partir da perspectiva CAN, são ineficientes em termos de consumo de energia e taxa de dados.

Esta tese define o estudo de uma arquitetura única (microcontrolador) para acoplar o protocolo CAN com o padrão IEEE 802.15.4 LR-WPAN, proporcionando assim, um módulo barato,

[4]**CSMA/CA** - é um método de transmissão que possui também mais parâmetros restritivos, o que contribui para a redução da ocorrência de colisões em uma rede (máquinas interligadas através de uma rede identificam uma colisão quando o nível de sinal aumenta no interior do cabo). Antes de transmitir efetivamente um pacote, a estação avisa sobre a transmissão e em quanto tempo a mesma irá realizar a tarefa. Dessa forma, as estações não tentarão transmitir, porque entendem que o canal está sendo usado por outra máquina, porém, o tempo que as máquinas esperam para que possam enviar seus pacotes não é indeterminado ou aleatório, as mesmas irão saber quando o meio estará livre.

de baixa potência, eficiente, seguro e compatível com muitas outras interfaces de redes e infra-estruturas existentes.

Alguns problemas envolvidos na criação do módulo proposto (CAN e IEEE 802.15.4) por possuir protocolos diferentes exige um grande esforço por parte do desenvolvedor para integrá-los. Estas redes são heterogêneas em vários aspectos, como ilustrado na Tabela 1.

Tabela 1 – Comparação da rede CAN com o IEEE 802.15.4

	Rede de Controle de Área CAN	**IEEE 802.15.4**
Camada Física	Barramento com terminadores	Spread spectrum RF
Camada MAC/LLC	Filtros de mensagens	Endereço de origem e destino
Camada MAC	Detecção de Colisão	Anticolisão
Camada de Rede/Topologia	Nós em paralelo, patrão-trabalhador	Ad-hoc, dispositivos de função completa, dispositivos de função reduzida
Segurança	Nenhuma	Listas de Controle de Acesso, Criptografia Simétrica.

Fonte: Adaptado de Kuban (2006).

Nas próximas seções deste trabalho, as limitações envolvidas na implementação e programação do módulo microcontrolador serão analisadas. A implantação deste sistema é mais provável de ocorrer em sistemas de controle e automação e sistemas críticos que necessitam de respostas em tempo real. Por isso, várias características essenciais, que afetam as respostas em tempo real, são investigadas. Estes incluem atrasos na troca de mensagens entre os nós da rede, memória utilizada para o armazenamento de dados na sincronização entre os protocolos CAN e o ZigBee, e ainda, os tempos nas operações de inserção e remoção de dados das estruturas de armazenamento.

1.1 JUSTIFICATIVA

A ausência de um padrão internacional de interfaceamento entre protocolos heterogêneos para comunicação sem fio fez com que engenheiros utilizassem diferentes tipos de interfaces e protocolos em diferentes ambientes de acordo com a sua necessidade, dificultando, assim, a interoperabilidade e futuras manutenções e comunicação do sistema. Muitos desses sistemas utilizam um nó de rede centralizado realizando a conversão dos protocolos e modificando a interface de comunicação.

É comum que a integração de protocolos diferentes se dê com a utilização de mais de um

microcontrolador. Assim, encontram-se trabalhos que para a integração de uma rede industrial com uma rede sem fio específica, utilizam-se de vários dispositivos microcontroladores que deem suporte para cada uma das tecnologias implementadas, assim, podendo fazer com que o custo operacional de implantação se torne alto e muitas vezes inviável.

1.2 OBJETIVOS

O objetivo deste trabalho é desenvolver um integrador de protocolos diferente usando apenas um microcontrolador que dê suporte na integração de um protocolo de rede industrial com um protocolo de rede sem fio. Para fins de comparação adotar-se-á o padrão de rede industrial CAN e o padrão de comunicação sem fio IEEE 802.15.4.

Este integrador de protocolos terá a capacidade de agregar vários sensores em uma área pré-determinada utilizando para controle dos nós, um protocolo de rede industrial e a comunicação entre nós sensores localizados em ambientes distantes através de um padrão sem fio.

1.3 ESTRATÉGIA DE DESENVOLVIMENTO

O objetivo principal da presente tese é o desenvolvimento de um nó de rede (integrador) capaz de integrar as tecnologias de comunicação de redes industriais com tecnologias de comunicação sem fios, independente de tecnologia ou protocolo de comunicação utilizado para o interfaceamento das tecnologias (*wireless e com fio*). O nó será capaz de processar esses dados e enviar para uma outra interface de comunicação da rede, provido com a mesma tecnologia do nó mestre/escravo.

No desenvolvimento do nó de rede, foi utilizado o ambiente de programação do Arduino e também o AVR Studio, permitindo ao projetista implementar os protocolos de comunicação, de acordo com as características do sistema, utilizando componentes pre-definidos. Para a implementação e testes do sistema foi utilizado o kit Arduino Uno R3, o módulo da ATMEL STK500, módulo de expansão STK501, rádio de comunicação da ATMEL RZ502, radio de comunicação XBee Pro e o placa de expansão CAN Shield para arduino. Todos os módulos utilizados possuem interfaces de comunicação para outros dispositivos e pinos de I/O. A linguagem de programação utilizada para a gravação dos algoritmos nos microcontroladores é a linguagem ANSI C/C++.

Além das características de integração de diferentes protocolos em um mesmo hardware, o nó de rede foi desenvolvido utilizando conceitos de estruturas de dados avançadas como, as

tabelas de dispersão (*Hash Tables*) e também conceitos de programação paralela (*threads*) para a sincronização dos protocolos de comunicação.

Tais características de hardware, oferecem suporte ao desenvolvimento de um nó de rede heterogêneo que agregue diferentes tecnologias de forma que possa ser aplicada em diversos ambientes em condições adversas.

Para os testes do nó de rede, foram desenvolvidos módulos Arduino providos de antenas de comunicação sem fio XBee baseadas no padrão IEEE 802.15.4 e placas para controle do protocolo CAN. O objetivo foi verificar a integridade dos dados e do nó de rede e também o quanto de memória a nova tecnologia consegue administrar utilizando o conceito de tabelas de dispersão. Também é analisado o tempo de comunicação sem fio e, a quantidade de pacotes perdidos na comunicação.

1.4 ORGANIZAÇÃO DO TEXTO

Além desta introdução geral no Capítulo 1, que contém a contextualização do projeto, um breve histórico do protocolo CAN, do padrão IEEE 802.15.4 e os trabalhos mais relevantes na área, o texto foi organizado em 8 Capítulos.

No Capítulo 2 são introduzidos os principais conceitos de Redes Industriais e a descrição do protocolo CAN, métodos de transferência de mensagens no barramento, as tecnologias de transmissão de dados utilizadas e as diretrizes para o desenvolvimento dos módulos para redes de sensores sem fio (RSSF).

No Capítulo 3 são apresentadas as características dos Principais Padrões de Comunicação sem Fio e o padrão IEEE 802.15.4, como formato de pacotes, abrangência de comunicação, tipos de erros, paradigmas da sua utilização na comunicação sem fio, contextos que se adéquam à elaboração do projeto.

No Capítulo 4 está conceituado o Tabelas de Dispersão, características e trabalhos na literatura que se apoiaram nesta técnica para otimização de memória e tempo de acesso à dados e, utilizadas nesta tese.

No Capítulo 5 são apresentadas as ferramentas que são utilizadas para os testes e implementação do módulo, suas características, análise de desempenho e onde se enquadram dentro do contexto do projeto.

No Capítulo 6 é apresentado o desenvolvimento do nó de rede com a integração da rede CAN com o padrão IEEE 802.15.4

No Capítulo 7 são realizadas as análises e resultados do trabalho. Estes testes que foram realizados para comunicação serial validam a funcionalidade do sistema.

No Capítulo 8 são apresentadas as considerações finais deste trabalho. Além disso, as contribuições da tese e possíveis trabalhos futuros a serem desenvolvidos.

2 REDES DE COMUNICAÇÃO INDUSTRIAL

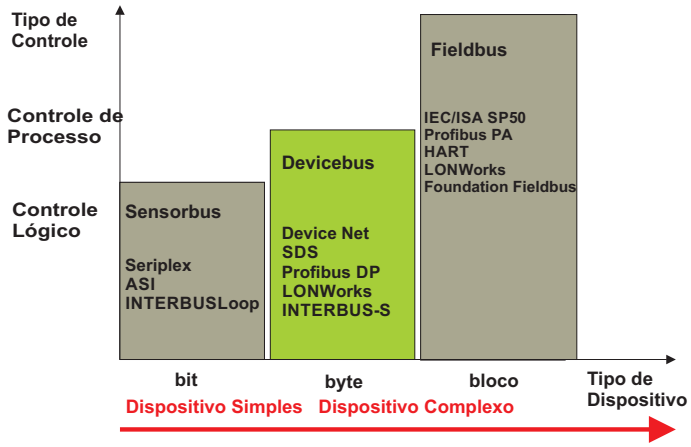
Para criar uma fundamentação teórica, este capítulo fornece uma visão geral do protocolo CAN e apresenta um breve estudo do formato dos pacotes e protocolos que este protocolo utiliza.

2.1 REDES INDUSTRIAIS

Uma rede de computadores é composta por um conjunto de computadores autônomos que estão interconectados entre si por um meio de transmissão qualquer, como: cabos metálicos, óticos ou sem fio (*wireless*) (TANENBAUM, 2003a).

Com a necessidade de realizar a comunicação entre computadores e os controladores lógicos programáveis (CLP) é que surgem as redes industriais. Assim, foi possível compartilhar recursos e dados coletados pela rede com maior segurança e criar sistemas de informação com maior autonomia na tomada de decisão (NOGUEIRA, 2009).Estas redes podem ser observadas na Figura 2.

Figura 2 – Redes de Comunicação Industrial



Fonte: Adaptado de Guedes (2005).

Os Sistemas de Informações Gerenciais (SIG) se tornaram essenciais para a tomada de decisão em diversos ramos da indústria e atividades em geral. A comunicação entre dispositivos industriais é um campo muito importante a ser estudado. A supervisão na linha de produção

e o gerenciamento da comunicação entre os equipamentos em tempo real são necessidades imprescindíveis atualmente (NOGUEIRA, 2009).

As redes de comunicação para a automação industrial eram isoladas, atualmente, precisam estar conectadas há várias outras redes, podendo assim, compartilhar informações com o intuito de aperfeiçoar à tomada de decisão através dos Sistemas de Informações Gerenciais, economizando tempo, além, de otimizar o uso de matéria prima.

Existe a necessidade de se utilizar sistemas de comunicação que consigam suportar alguns requisitos típicos como: ambientes hostis, interferências eletromagnéticas, características de tempo real e volume de informação trocada, o que obriga a constante busca por novas técnicas e meios de estabelecer essa comunicação (NOGUEIRA, 2009).

Com isso, o desenvolvimento de aplicações, que dependendo da regra de negócios a ser analisada conseguem aumentar a agilidade e a eficiência, e que são capazes de gerenciar, supervisionar, controlar e proteger as informações das redes industriais não podem ser descartadas para seja garantida a integridade dos dados na comunicação. Assim, sistemas de controle baseados em protocolos de comunicação digital tem crescido nas mais variadas plantas industriais (CHEN; MOK, 2001). Os Sistemas de Informações Gerenciais integra a Tecnologia de Automação (TA) e a Tecnologia de Informação (TI) possibilitando a tomada de decisão dentro das empresas.

Para que exista a comunicação em uma rede é necessário que esta siga as especificações (protocolos) do modelo de referência OSI (*Open System Interconection*). O padrão do modelo OSI é a padronização na comunicação entre equipamentos de fabricantes diferentes, fazendo com que os dispositivos integrados ao sistema possam trocar informações corretamente na rede.

2.2 CARACTERIZAÇÃO DAS REDES INDUSTRIAIS

O desejo controlar os processos industriais é pretendido pelos homens desde automação dos primeiros trabalhos através de algoritmos. Antes da explosão computacional, os processos eram operados manualmente por um grande número de trabalhadores, que utilizavam-se de instrumentos mecânicos elementares que possibilitavam o controle total do processo (GUTIERREZ; PAN, 2008).

Automação é um método que realiza processos automáticos que comandam e controlam os mecanismos para seu próprio funcionamento. Assim, automação busca caracterizar a participação de computadores no controle automático na indústria (MORAES; CASTRUCCI, 2007).

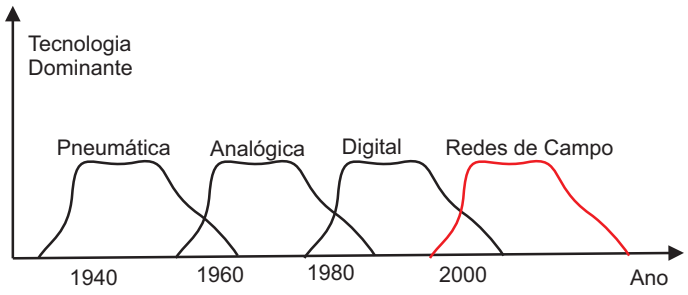
De acordo com Weg (2002) a automação envolve um conjunto de técnicas de controle, onde é criado um sistema ativo, capaz de tomar a decisão adequada em função das informações que este coleta do processo em que está atuando. Assim, o sistema de informação pode tomar a melhor decisão dependendo da informação analisada .

Os sistemas de informações foram desenvolvidos de forma a orientar qual a melhor decisão a ser tomada, além de eliminar/diminuir possíveis riscos envolvidos na fase produção ou coleta de dados. Tarefas que antes implicavam em alto risco para operadores de equipamentos podem ser realizadas remotamente sem qualquer risco. Estas técnicas são aplicadas em várias áreas da engenharia para buscar a otimização dos processos industriais.

De acordo com Nogueira (2009), os sistemas de controle de processo e manufatura surgiram em meados de 1940 (Figura 3), foram baseados em tecnologia mecânica e pneumática em sua primeira faze histórica. Na segunda fase foram substituídas por sinais elétricos analógicos, fase que ganhou grande impulso nos anos 50, com o surgimento dos controladores eletrônicos (ECU), que permitiam que os dados pudessem trafegar por distâncias maiores nos meios de transmissão.

Para o autor Djiev (2003), entre as décadas de 60 e 70, a utilização de microcontroladores e computadores para solucionar problemas de controle eram inviáveis. Com o barateamento do hardware, os computadores passaram a ser utilizados em todos os campos da indústria, desde o nível de processo até o nível de administração da empresa. A evolução das tecnologias de comunicação podem ser observadas na Figura 3.

Figura 3 – Evolução das Tecnologias de Comunicação



Fonte: Adaptado de Guedes (2005).

Na indústria o objetivo da implantação de processos automáticos é de alcançar uma maior produtividade com a minimização dos custos. A implantação de sistemas automatizados possui um custo elevado e, além disso, as instalações requerem recursos para sua manutenção. O principal motivo da automação é aumentar a qualidade dos processos, reduzindo perdas, e pos-

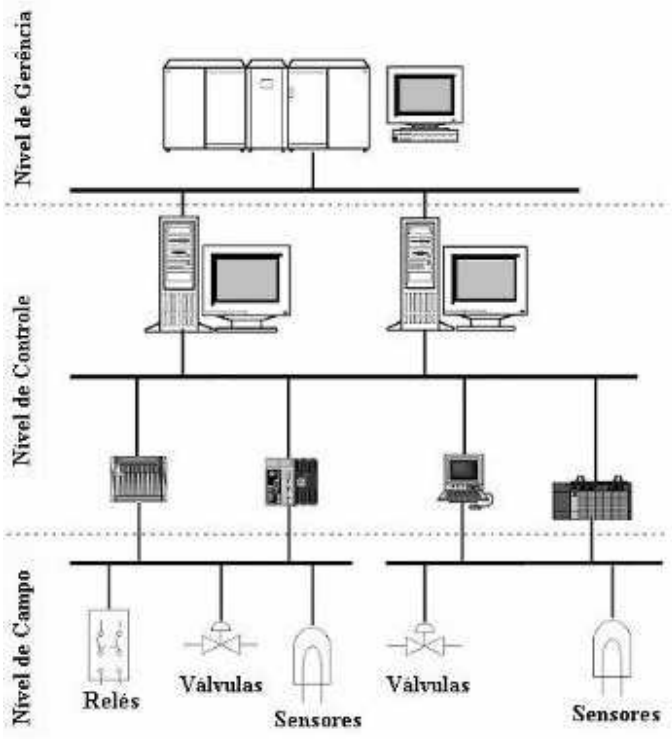
sibilitando a fabricação de bens que de outra forma não poderiam ser produzidos. A garantia da segurança é outro aspecto que consome grandes volumes de investimentos para a automação de processos industriais e de infra-estrutura críticas (GUTIERREZ; PAN, 2008).

2.3 CLASSIFICAÇÃO DAS REDES INDUSTRIAIS

Atualmente descreve-se os diversos sistemas que coordenam o processo produtivo através de modelos conceituais. Devido à complexidade destes sistemas é comum estruturá-los em níveis hierárquicos para facilitar a compreensão. Cada nível hierárquico tem associado um nível de comunicação com exigências próprias na rede (NOGUEIRA, 2009).

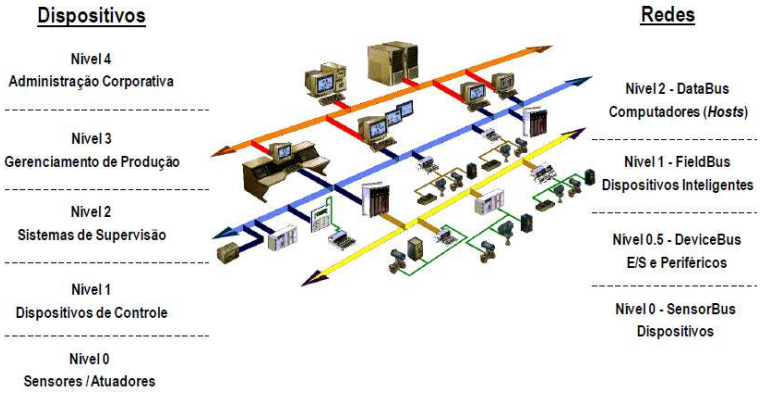
Na Figura 4 é evidenciado uma estrutura de hierarquias que é composta, com os níveis básicos para as empresas: Nível de Campo, Controle e Gerência. Na Figura 5 ilustra-se uma estrutura funcional, com as divisões Instrumentação (sensores e atuadores), Redes de dispositivos discretos, Controladores, Supervisão e Gestão da Produção.

Figura 4 – Classificação das Redes Industriais



Fonte: (GUEDES, 2005).

Figura 5 – Classificação das Redes Industriais e Estrutura Funcional.



Fonte: (GUEDES, 2005).

A rede **Sensorbus** é geralmente usada para conectar pequenos sensores, também é usada para interligar equipamentos simples e pequenos diretamente à rede. A rede Sensorbus é composta por sensores e atuadores de menor valor. Esse tipo de rede busca minimizar os custos de conexão (MONTEZ, 2005 apud NOGUEIRA, 2009). O tempo de latência da rede é da ordem de milissegundos, para distâncias de até 200 metros (BORGES, 2008). São exemplos de redes Sensorbus as redes Seriplex, ASI (*Automation Systems Interconnect*) e CAN (NOGUEIRA, 2009).

A Rede **Devicebus** está entre as redes Sensorbus e Fieldbus. Podem alcançar até 500m de distância. Nesta rede é permitido transferir blocos em prioridade menor se comparado aos dados no formato de *bytes*. Possui características de transferência rápida (dezenas de milissegundos) de dados como da rede Sensorbus, gerenciando uma maior quantidade de dados e equipamentos (MONTEZ, 2005 apud NOGUEIRA, 2009). Pode-se destacar as redes DeviceNet, Profibus DP (*Decentralized Peripherals*), entre outras (NOGUEIRA, 2009).

Na rede **Fieldbus** conecta-se equipamentos de E/S além de cobrir distâncias que podem chegar a 10 Km. Os equipamentos interconectados nessa rede desempenham funções específicas como o controle de fluxo de informações e processos. Os tempos de comunicação são longos (centenas de milissegundos) (MONTEZ, 2005 apud NOGUEIRA, 2009). São exemplos as redes Modbus Plus, Profibus FMS (*Fieldbus Message Specification*) (NOGUEIRA, 2009).

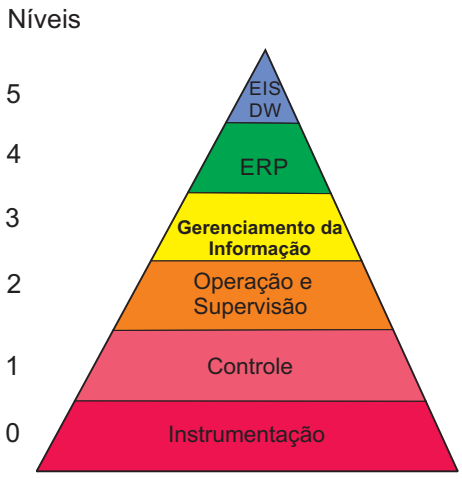
A rede **Databus** possibilita a comunicação entre os sistemas de supervisão e os sistemas de gestão (Sistemas de Informações Gerenciais - SIG). Os tempos latência para a comunicação podem ser de alguns segundos e de até minutos, podem alcançar distâncias de até 100 Km. Os dados que trafegam na rede são arquivos com grande volume de informação. Utiliza rede Ethernet (LAN (*Local Area Network*), WAN (*Wide Area Network*), Internet) (MONTEZ, 2005

apud NOGUEIRA, 2009).

De acordo com Nogueira (2009), o nível de gerenciamento da produção é o nível superior da pirâmide e está reservado aos sistemas de informação. Estes sistemas destinam-se à gestão global da empresa. Como é um ambiente voltado mais para a Tecnologia da Informação (TI) prioriza a confidencialidade dos dados na rede, protegendo contra acessos não autorizados, assim como a integridade e a disponibilidade dos mesmos.

Segundo Filho e Finkel (2003), uma maneira simples e didática de visualizar toda essa estrutura descrita anteriormente pode ser expressa na Figura 6:

Figura 6 – Pirâmide Hierárquica detalhada



Fonte: (GUEDES, 2005)

Este modelo hierárquico divide os sistemas de manufatura em vários níveis (NOGUEIRA, 2009):

- Nível 0: Instrumentação;
- Nivel 1: Controladores: CLPs, sistemas digitais remotos de controle distribuído (SDCDs), etc.;
- Nivel 2: Supervisão: Sistemas de supervisão e aquisição de dados (SCADA), interface homem maquina (IHM) e otimizadores de processo dentro do conceito de APC (*Advanced Process Control*);
- Nivel 3: Gestão da produção: Sistemas MES (*Manufacturing Execution System*), PIMS (*Process Information Management System*), APS (*Advanced Planning and Scheduling*),

LIMS (*Lab Information System*), Sistemas de Manutenção (MMS - *Maintenance Management System*), Sistema de Gestão de Ativos (AMS - *Asset Management System*), etc.;

- Nivel 4: Sistemas Integrados de Gestão Empresarial (ERP - *Enterprise Resource Planning*);
- Nivel 5: *Data Warehousing* (DW) corporativos, um sistema de computação utilizado para armazenar informações relativas às atividades de uma organização em bancos de dados e Sistemas EIS (*Executive Information Systems*), que tem como objetivo principal dar suporte à tomada de decisão.

Para entender o modelo proposto pela Figura 6 basta compreender que no nível 3 ou acima é onde são utilizados os *softwares* gerenciais e corporativos, conectados pela rede de computadores, permitindo a comunicação entre todos os pontos da rede para o gerenciamento industrial.

No nível 2 é necessário interligar as estações de operação a estações de cálculo, banco de dados para que seja possível realizar funções de supervisão, armazenamento e tratamento das informações do processo (NOGUEIRA, 2009).

O nível 1 tem por função conectar os CLPs e as estações de controle e o nível 0 faz a ponte de comunicação entre os controladores e aos dados dos equipamentos e componentes do processo (NOGUEIRA, 2009).

Cada um dos níveis possuem necessidades diferentes para a configuração e instalação da rede assim, existe uma enorme variedades de redes que podem atuar em cada dos níveis da pirâmide.

Por isso é necessário conhecer o tipo de aplicação que o usuário final está procurando para assim utilizar uma tecnologia que seja compatível e que possa oferecer um melhor desempenho e consequentemente menos falhas no sistema (FORTE, 2004).

Dentre os tipos de redes industriais existentes, para a execução desta tese, adotou-se a rede industrial CAN para a comunicação nos nós dentro da rede, entretanto, poderia ser utilizada qualquer outra rede industrial para o desenvolvimento da proposta.

2.4 O PADRÃO CAN - *Controller Area Network*

Para supervisionar o controle e a aquisição de dados (SCADA) são utilizadas as redes *Foundation Fieldbus* (WILD; PEREIRA, 1999), a *Profibus* (POPP, 1997) e a CAN. O protocolo

CAN apresenta-se como uma alternativa viável para o desenvolvimento de sistemas de automação industrial. A utilização de objetos distribuídos em conjunto com os outros barramentos industriais traz algumas dificuldades relacionadas como no modelo de controle geralmente utilizado por eles. O barramento *Fundation Fieldbus*, por exemplo, apesar de permitir o uso de modelos cliente/servidor e "*publisher/subscriber*", utiliza-se do conceito de blocos funcionais padronizados que dificulta a utilização de orientação a objetos no mesmo. Entretanto, com relação ao *Profibus*, tem-se que a utilização de mestres e escravos com comunicação ponto a ponto restringe o uso de objetos distribuídos, uma vez que a autonomia dos mesmos é uma das características que se deseja preservar.

Diferentemente da comunicação ponto a ponto geralmente utilizada pelos outros barramentos, o CAN possui a característica de *broadcast*[1] que pode ser convenientemente explorada pelos sistemas de automação industrial. Além disso, a priorização de mensagens utilizada no barramento CAN oferece algumas propriedades de tempo-real que podem ser convenientemente aproveitadas. O barramento CAN oferece ainda a possibilidade de se definir um protocolo de alto nível para as aplicações, de acordo com as necessidades do usuário, uma vez que o seu padrão básico cobre somente as duas primeiras camadas do padrão ISO[2]/OSI (BRUDNA, 2000).

Assim, quanto à utilização de objetos distribuídos, tem-se que o barramento CAN apresenta-se como uma alternativa bastante adequada, uma vez que este oferece muita liberdade ao projetista, para a utilização de qualquer modelo de controle e padrão de comunicação desejado. Por último, seu custo é suficientemente baixo para ser utilizado em micro controladores, o que favorece o desenvolvimento de sistemas de automação com controle distribuído.

A rede CAN teve amplo sucesso e popularidade em muitas e variadas aplicações. Inicialmente desenvolvida para ser utilizada em barramentos de instrumentação de automóveis, a sua utilização tem evoluído ao longo da última década, para incluir aplicações tão variadas como construção e automação HVAC (*Heating, Ventilation and Air Conditioning*), sistemas de controle de fábricas e dispositivos médicos.

2.5 DESCRIÇÃO DO PROTOCOLO CAN

O barramento CAN é um protocolo de comunicação serial desenvolvido inicialmente para aplicações automotivas que, recentemente, vem sendo utilizado em sistemas de automação industrial. O protocolo CAN é baseado na técnica de CSMA/CR (*Carrier Sense Multiple Access/-*

[1] **Broadcast** é o processo pelo qual se transmite ou difunde determinada informação, tendo como principal característica que a mesma informação está sendo enviada para muitos receptores ao mesmo tempo.

[2] Organização Internacional para Padronização - *International Standardization Organization*.

Collision Resolution), às vezes, também chamado de CSMA/CD+AMP (*Carrier Sense Multiple Access/Collision Detection and Arbitration on Message Priority*), de acesso ao meio de transmissão. Isto significa que sempre que ocorrer uma colisão entre duas ou mais mensagens, a de mais alta prioridade terá o acesso ao meio físico assegurado e prosseguirá a transmissão.

As características básicas dos pacotes que trafegam pelo barramento CAN são: 8 *bytes* de dados, velocidade de até 1Mbit/s, priorização das mensagens, recepção *multicast*[3] com sincronização, detecção de erros e sinalização e retransmissão automática de mensagens corrompidas. Todas estas características propiciam simplicidade, alta confiabilidade e segurança, além de baixo custo. A primeira versão do protocolo CAN 2.0A (BOSCH, 1991) especificava somente mensagens do tipo padrão com identificadores de 11 bits enquanto que o CAN 2.0B (BOSCH, 1991) admite também mensagens estendidas com identificadores de 29 bits. O protocolo CAN foi adotado em 1993 com o padrão mundial ISO11898, pela *International Standardization Organization*.

O protocolo é usualmente implementado em um controlador na forma de um circuito integrado, mas também podem ser encontrados no mercado microcontroladores de 8, 16 e 32 bits com controladores CAN integrados. Os controladores CAN e os microcontroladores com controladores CAN integrados são fabricados por um grande número de indústrias tais como *Intel*, *Motorola*, *Philips*, *Siemens* e *Texas Instruments*, resultando em torno de 50 circuitos integrados de 15 fabricantes diferentes. A utilização do protocolo CAN, na indústria automobilística, resultou numa produção em grande escala de controladores CAN, atualmente na casa dos milhões ao ano (BRUDNA, 2000).

A afirmação e o crescimento do protocolo CAN está fortemente baseado na organização dos fabricantes em torno de associações tais como a CiA (*CAN in Automation*) assim como na existência de uma série de ferramentas de software para o desenvolvimento, simulação, configuração e monitoração de aplicações bem como na disponibilidade de hardware na forma de placas de controle, placas ISA (*Industry Standard Architeture*), PCI (*Peripheral Component Interconnect*) e outras (BRUDNA, 2000).

2.6 CARACTERÍSTICAS GERAIS DO BARRAMENTO CAN

De acordo com o modelo OSI/ISO, o padrão CAN 2.0B, apresentado na Tabela 2, é constituído somente de duas camadas: a Camada de Dados (*Data Link Layer*) e a Camada Física (*Physical Layer*). As características do barramento CAN podem ser resumidas por Brudna

[3] **Multicast** é a entrega de informação para múltiplos destinatários simultaneamente.

(2000):

- Baseado no conceito de *broadcast*;
- Um esquema de arbitragem não destrutiva (*bitwise arbitration*) descentralizada, baseada na adoção dos níveis "dominante" e "recessivo", é usado para controlar o acesso ao barramento;
- As mensagens de dados são pequenas (no máximo oito *bytes* de dados) e são conferidas por *checksum*;
- Não há endereço explícito nas mensagens. Em vez disso, cada mensagem carrega um identificador que controla sua prioridade no barramento e que pode servir como uma identificação do conteúdo da mesma;
- Utiliza um elaborado esquema de tratamento de erros que resulta na retransmissão das mensagens que não são apropriadamente recebidas;
- Fornece meios efetivos para isolar falhas e remover nós com problemas do barramento;
- Oferece meios para filtragem das mensagens;
- O meio físico de transmissão pode ser escolhido de acordo com as necessidades. O mais comum é o par trançado, mas também podem ser utilizados outros meios de transmissão tais como fibra ótica e rádio frequência.

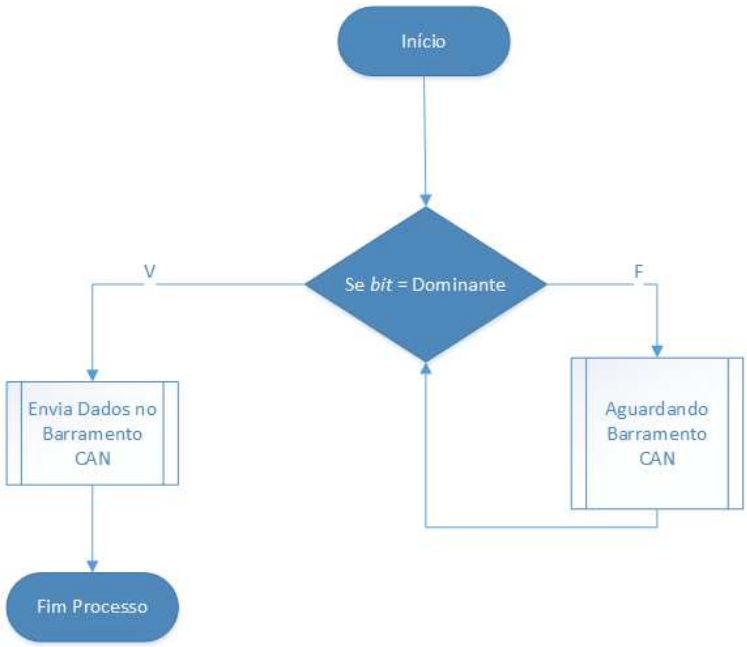
Tabela 2 – Camadas do padrão CAN 2.0B

Camadas - OSI	**Função**	**Especificação**
7	Aplicação	Especificado pelo projetista.
6	Apresentação	Vazio
5	Sessão	Vazio
4	Transporte	Vazio
3	Rede	Vazio
2	Link de Dados	Coberto pelo CAN e padrão ISO.
1	Física	Coberto pela ISO e parcialmente pelo CAN.

Uma das propriedades mais importantes do barramento é o esquema de arbitração binária (*bitwise arbitration*) que fornece uma maneira de resolver colisão de mensagens. Sempre

que dois nós começarem a transmitir mensagens ao mesmo tempo, o mecanismo de arbitragem garante que a mensagem de mais alta prioridade será enviada. Isto é conseguido através da definição de dois níveis de barramento chamados “recessivo” e “dominante”. Um nível “dominante” sempre sobrescreve um nível “recessivo”. Assim, enquanto um nó está enviando uma mensagem, ele compara o nível do bit transmitido com o nível monitorado no barramento. Se um nó tenta enviar um nível “recessivo” e detecta um “dominante”, ele perde a arbitragem e interrompe o processo de transmissão (Figura 7).

Figura 7 – Fluxograma de Acesso ao Barramento CAN



Fonte: Próprio Autor.

A taxa de transmissão, por sua vez, é limitada pelo comprimento do barramento utilizado. No caso do uso de par trançado, por exemplo, temos uma determinada relação entre a taxa de transmissão e o comprimento máximo do barramento, como mostra a Tabela 3. Essa limitação decorre da necessidade de sincronização entre as estações componentes do barramento a qual dá suporte ao esquema de arbitração binária.

2.7 QUADROS DE UMA MENSAGEM CAN

O barramento CAN na sua versão 2.0B é formado por quatro tipos de quadros (*frames*) para controlar a transferência de mensagens (BOSCH, 1991; BRUDNA, 2000):

- Quadro de Dados (*DataFrame*);

Tabela 3 – Taxa de transmissão e comprimento do barramento

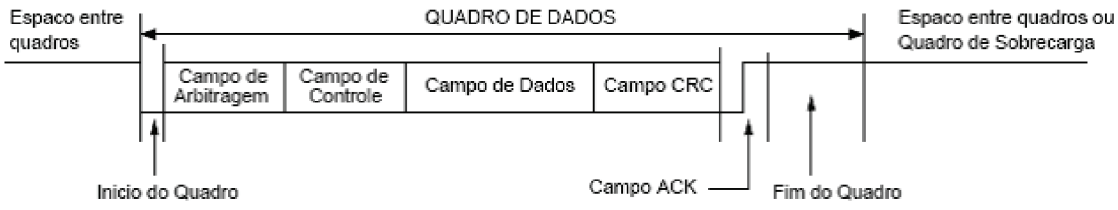
Taxa (Kbit/s)	Distância Máxima (m).
1000	40
500	130
250	270
125	530
100	620
50	1300
20	3300
10	6700
5	10000

- Quadro Remoto (*RemoteFrame*);
- Quadro de Erro (*ErrorFrame*);
- Quadro de Sobrecarga (*Overload Frame*).

2.7.1 Definição do quadro de dados do barramento CAN

O Quadro de Dados leva dados do transmissor para os receptores e é composto de sete campos diferentes mostrados na Figura 8 (BOSCH, 1991; BRUDNA, 2000; HUBERT, 2001):

Figura 8 – Quadro de Dados



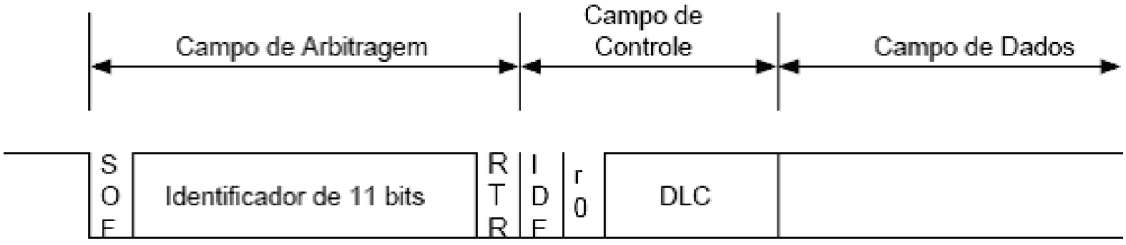
Fonte: Próprio Autor

- **Início do quadro (SOF - *Start of Frame*):** marca o início do quadro com um bit "dominante" simples.
- **Campo de arbitragem:** no formato padrão (Figura 9), consiste de um identificador de 11 *bits* e do *bit* RTR (*Remote Transmission Request*), o qual indica se é um Quadro de Dados (RTR - "dominante") ou Quadro Remoto (RTR - "recessivo"). No formato

estendido (Figura 10), é formado por um identificador de 29 *bits*, do bit SRR - *Substitute Remote Request* (“recessivo”, substitui o *bit* RTR do formato padrão), *bit* IDE (identifica se o quadro é padrão, IDE - “dominante”,ou estendido, IDE - “recessivo”) e pelo *bit* RTR.

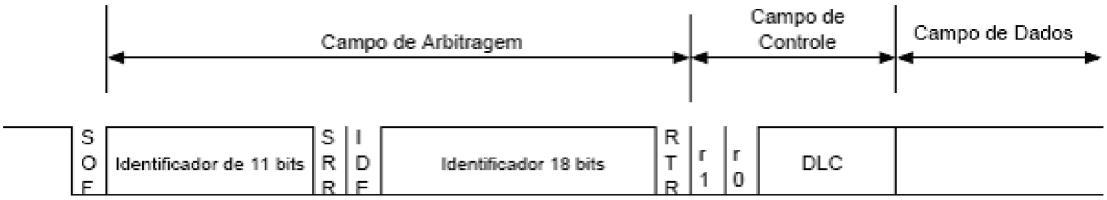
- **Campo de controle:** no formato padrão é constituído do DLC (*Data Length Code*), que indica o número de *bytes* no campo de dados, e dos *bits* IDE e r0 (reservado). No formato estendido, é formado pelo DLC e pelos *bits* reservados r1e r0.
- **Campo de dados:** consiste dos dados a serem transmitidos pelo Quadro de Dados.
- **Campo CRC:** contém a sequência de CRC (*Cyclic Redundancy Code*) de 15 *bits* seguida por um delimitador (1 *bit*).
- **Campo ACK (*Acknowledgement*):** é composto pelo *ACK Slot* (1 *bit*) e Delimitador de ACK (1 *bit*). É utilizado pelos nós receptores, para indicar o correto recebimento de uma mensagem.
- **Fim do quadro:** É formado por uma sequência de sete *bits* “recessivos” e serve para delimitar o quadro.

Figura 9 – Quadro de Dados Padrão



Fonte: Próprio Autor

Figura 10 – Quadro de Dados Estendido

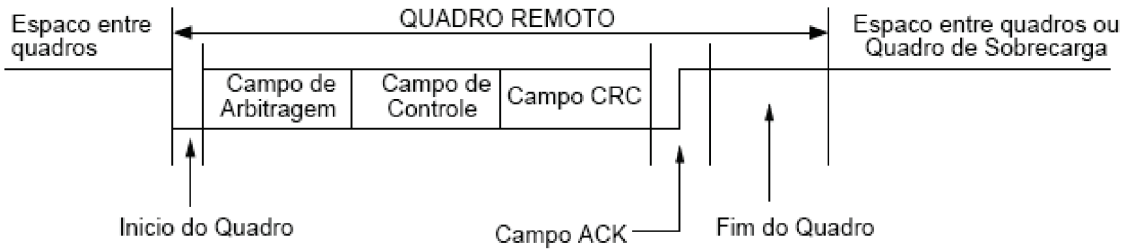


Fonte: Próprio Autor

2.7.2 Quadro remoto do barramento CAN

O Quadro Remoto, mostrado na Figura 11 é utilizado para requisitar dados. É formado pelos campos: Início do Quadro, Campo de Arbitragem, Campo de Controle, Campo de CRC, Campo de ACK e Fim do Quadro. No Quadro Remoto o *bit* RTR é "recessivo" e não há campo de dados. Este tipo de quadro é utilizado quando um nó deseja solicitar dados a outro nó. O nó que possuía informação responde, então, através de um Quadro de Dados com o mesmo identificador da solicitação (BOSCH, 1991; BRUDNA, 2000; HUBERT, 2001).

Figura 11 – Quadro Remoto



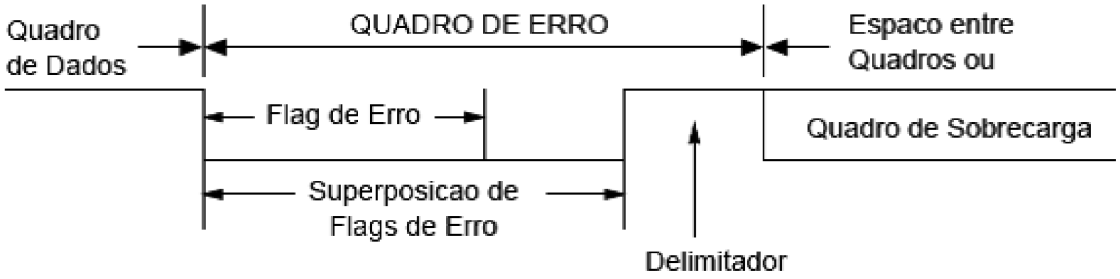
Fonte: Próprio Autor

2.7.3 Quadro de erro do barramento CAN

Um Quadro de Erro é transmitido por um nó sempre que for detectado algum tipo de erro no barramento. Se um nó detectar um problema comum Quadro de Dados, por exemplo, isto será imediatamente indicado por um Quadro de Erro. O Quadro de Erro, como mostra a Figura 12, é composto por dois campos (HUBERT, 2001):

- **Flags de Erro:** é formado pela superposição de *Flags* de Erros provenientes de diferentes nós. O comprimento total pode variar de seis a doze *bits*. Existem dois tipos de *Flags* de Erro: *Flags* de Erro Ativos, que consistem de seis *bits* "dominantes" consecutivos, e *Flags* de Erro Passivos, constituídos por seis *bits* "recessivos" consecutivos.
- **Delimitador de Erro:** consiste de oito bits "recessivos".

Figura 12 – Quadro de Erro



Fonte: Próprio Autor

2.7.4 Quadro de sobrecarga (*Overload Frame*) do CAN

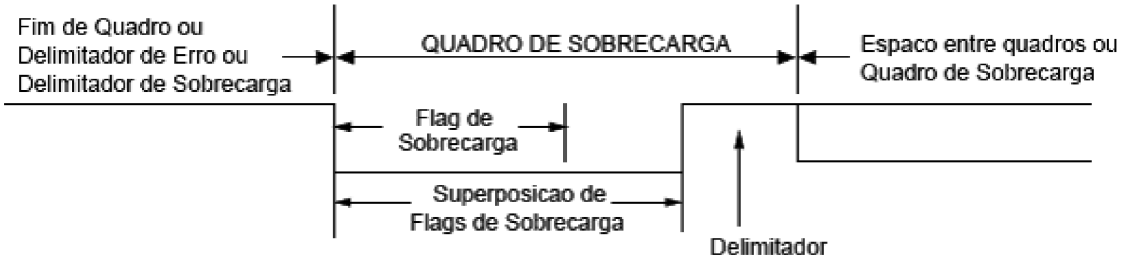
Existem duas condições que levam à transmissão de um Quadro de Sobrecarga:

- As condições internas de um receptor, o que requer um atraso até o próximo Quadro de Dados ou Quadro Remoto.
- Detecção de um *bit* "dominante" no espaço entre dois quadros.

No máximo dois Quadros de Sobrecarga podem ser gerados para atrasar o próximo Quadro de Dados ou Quadro Remoto. O Quadro de Sobrecarga, mostrado na Figura 13, é formado pelos campos:

- ***Flag* de Sobrecarga:** consiste de seis *bits* "dominantes".
- **Delimitador de Sobrecarga:** consiste de oito *bits* "recessivos".

Figura 13 – Quadro de Sobrecarga

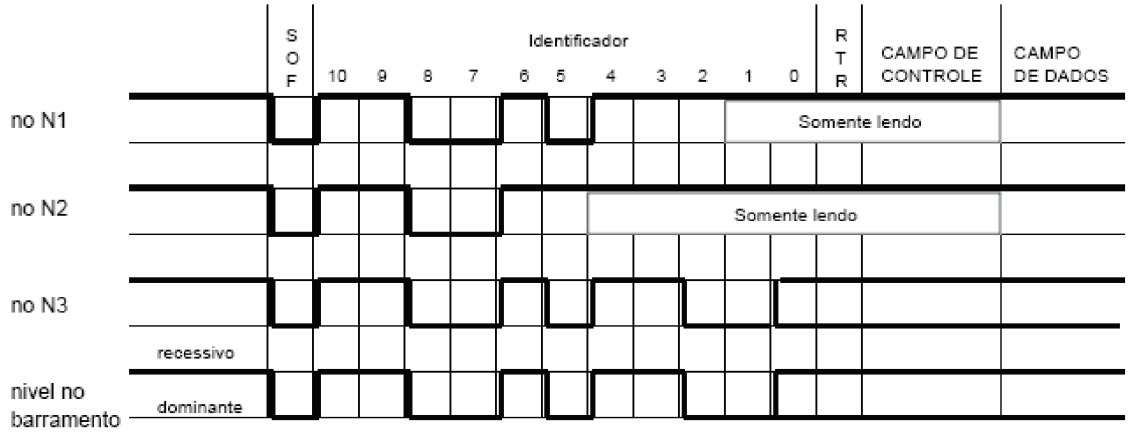


Fonte: Próprio Autor

2.8 ARBITRAGEM DO BARRAMENTO CAN

Utiliza o processo de dominância de bits para garantir que a prioridade entre as mensagens venha a ser mantida. O processo de arbitragem ocorre sempre que dois ou mais nós iniciam uma transmissão exatamente ao mesmo tempo, após verificar que o barramento está desocupado. Um exemplo deste processo, onde se tem três estações N1, N2 e N3 conectadas a um barramento, pode ser visto na Figura 14. Neste exemplo, vê-se que nada acontece até o *bit* cinco, onde os nós N1 e N3 transmitem *bits* "dominantes" ("0") e o nó N2 tenta transmitir um *bit* "recessivo". Por ter escrito um *bit* "recessivo" e lido um *bit* "dominante" no barramento, o nó N2 perde a arbitragem e passa somente a ler o barramento, emitindo *bits* "recessivos". Finalmente, no *bit* dois, o nó N1 perde a arbitragem para o nó N3, que possui um identificador com valor menor e, por isso, com maior prioridade do que os nós N1 e N2 (BRUDNA, 2000; HUBERT, 2001).

Figura 14 – Processo de Arbitragem no Barramento CAN



Fonte: Próprio Autor

O resultado deste processo de arbitragem é que a mensagem com a prioridade mais alta sempre consegue ser transmitida sem atrasos. Os nós N1 e N2, cujas mensagens perderam o processo de arbitragem, terão suas mensagens transmitidas depois que o nó três finalizar a sua transmissão.

2.9 DETECÇÃO E SINALIZAÇÃO DE ERROS

São utilizados dois mecanismos de detecção de erros no nível de *bits*: monitoração e inserção de *bits* (*bit stuffinng*). No nível de mensagens, são usados o *Cyclic Redundancy Check* (CRC) e checagem dos quadros (*Frame check*). A sinalização de qualquer erro detectado é feita

através de um Quadro de Erro.

A monitoração simultânea, por todos os nós, das mensagens que circulam no barramento permite a detecção de mensagens corrompidas. Desta maneira, os campos das mensagens são constantemente verificados de acordo com o tipo da mensagem, assim como o seu CRC. Caso algum nó detecte problemas com uma mensagem, ele indicará isso através de um Quadro de Erro. Tão logo seja possível, a mensagem será retransmitida garantindo, assim, que todas as mensagens serão corretamente recebidas por todos os nós.

Há também mecanismos de proteção contra nós com algum tipo de falha. Dessa maneira, se um dos nós estiver constantemente indicando o recebimento de mensagens corrompidas, ele estará perturbando o barramento com sucessivas retransmissões de mensagens. Assim, um contador interno de erros existente em cada controlador CAN, força o nó a se desconectar quando um certo número de erros é atingido.

2.10 FILTRAGEM DE MENSAGENS

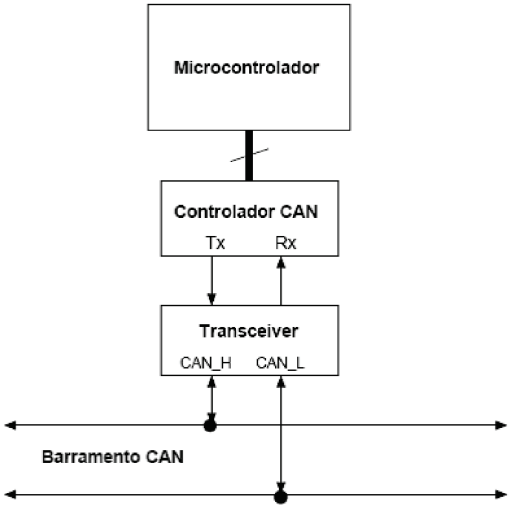
Como dito anteriormente, o protocolo CAN é implementado na forma de um controlador que, usualmente, também se comunica com um microcontrolador. Assim, a maioria dos controladores de protocolo oferece um serviço de filtragem de mensagens que faz com que somente as mensagens como padrão de identificação pré-programado sejam armazenadas e sinalizadas no microcontrolador. Isso possibilita uma economia de tempo de leitura e processamento das mensagens recebidas, liberando o microprocessador para tarefas mais importantes, contanto que a filtragem seja corretamente configurada. Essa operação normalmente envolve a configuração de duas máscaras para o identificador das mensagens, de forma a selecionar as mensagens ou grupo de mensagens desejadas e descartar as não desejadas.

2.11 INTERFACE COM MICROCONTROLADOR

Um nó é geralmente conectado a um barramento constituído de dois fios terminados em ambas as pontas por dois resistores. O controlador CAN pode estar diretamente conectado ao microcontrolador ou, como dito anteriormente, o microcontrolador pode possuir um controlador CAN internamente. A ligação do nó com o meio físico normalmente, se dá através de um *transceiver*, o qual se comunica serialmente com o controlador através do pino de saída Tx e do pino de entrada Rx. A função do *transceiver*, no caso do uso de par trançado, é transformar os *bits* que entram e saem do controlador em tensão diferencial a ser aplicada ao barramento.

O modo de transmissão diferencial é utilizado para proporcionar imunidade a interferências eletromagnéticas ao barramento. A alimentação do *transceiver* é feita com cinco volts e o nível "recessivo" corresponde a uma diferença de tensão menor do que 0,5V entre CAN_H e CAN_L, com a tensão em CAN_H sendo normalmente maior do que CAN_L. O nível "dominante" é detectado quando a diferença de tensão for de no mínimo 0,9V sendo que a tensão nominal neste estado é de 3,5V para CAN_H e 1,5V para CAN_L. Convém lembrar que os sinais dos pinos Tx e Rx possuem direções definidas enquanto que os sinais CAN_H e CAN_L não. O esquema deste tipo de conexão é mostrado na Figura 15.

Figura 15 – Esquema para a interface do microcontrolador com o barramento CAN



Fonte: Próprio Autor

2.12 CAMADAS DO PROTOCOLO CAN

Embora a especificação original refere-se às Camadas de Objeto e de Transferência (TCP), as mais recentes implementações rompem os protocolos para a Camada Física (PHY) e para a Camada de Dados, com a Camada de Dados dividida para as camadas de Controle Lógico (LLC - *Logical Link Control*) e (MAC - *Medium Access Control*) Controle de Acesso ao Meio (HUANG, 2003). A Tabela 4 mostra a organização e estrutura das camadas do CAN e as responsabilidades das camadas associadas.

Tabela 4 – Camadas CAN e respectivas tarefas

Camada	Todos os Nós	Nós Supervisores
Aplicação (APP)	Programas de Usuário	
Controle de Link Lógico (LLC)	Filtro de identificação, notificação de sobrecarga, gestão de recuperação.	Falha
Controle de Acesso ao Meio (MAC)	Estrutura de codificação e formatação, detecção e sinalização de erro, confirmação.	Falha
Física (PHY)	Tempo de bit, sincronização de codificação e decodificação de bits.	Gestão de falha no barramento

2.12.1 Camada física do CAN

A camada física trata de sincronização, codificação, bem como a ligação física entre dois nós de rede (KUROSE, 2005). A camada física do protocolo CAN não foi especificada no modelo 2.0, mais tarde, foi incorporada na especificação ISO11898 (STANDARD, 1993). Embora outros métodos de conexão sejam utilizados, o meio físico predominante para a comunicação é um simples par trançado, encerrado em cada extremidade com um resistor 124 Ohm. A especificação descreve o estado lógico do barramento como dominante e recessivo, em que o estado é o dominante com valor lógico "zero", que leva o barramento para o seu estado zero, independente de quantos lógicos "1" estão no barramento. Este mecanismo é utilizado para fornecer prioridade ao barramento mestre sobre nós subordinados e para a arbitragem entre os nós de igual valor. Se o barramento estiver livre, qualquer nó pode começar a transmitir uma nova mensagem. Se forem tentadas transmissões simultâneas, o conflito no barramento é resolvido usando os bits do campo identificador da mensagem. Todas as mensagens são recebidas no barramento (mas não são necessariamente transformados) por todos os nodos conectados a ele. Cada nó reconhece mensagens consistentes (correto) e mensagens inconsistentes. Isso garante a confiabilidade da ligação, fornecendo uma confirmação para o remetente. Em muitos dos casos, as camadas inferiores do CAN são implementadas em hardware com interrupções lógicas integradas que permitem a um modo de "*sleep*" e com "*wakeup*" automático, quando a atividade no barramento recomeça.

A taxa de dados do CAN é condicionada por vários fatores: comprimento do barramento,

frequência do oscilador e tolerância, velocidade do circuito condutor. Neste projeto, a taxa de dados exigidos pelo CAN cai bem abaixo do máximo de 1 Mbps, devido à taxa máxima do padrão IEEE 802.15.4 (250kbps). No entanto, ainda é benéfica para fornecer a transmissão dos parâmetros adequados (HUANG, 2003).

A taxa máxima de bits permitida pelo protocolo CAN é de 1 Mbps e diminui conforme o aumento do cabo. Em geral, para ambientes com baixo nível de ruído e taxa de dados, os critérios de amostragem podem ser relaxados e a tolerância do oscilador aumentada. Conforme o aumento da taxa de dados ou o aumento dos níveis de ruído, a amostragem é exigida normalmente e a tolerância do oscilador deve ser observada.

2.12.2 Camada de controle de acesso ao meio - CAN

A camada MAC é responsável pela codificação, formatação dos quadros de dados, detecção e sinalização de erros, e reconhecimentos. Cada nó ativo no barramento realiza acompanhamento, controle de redundância cíclica (CRC), "*Bit-stuffing*" e o controle dos quadros das mensagens. Erros globais e locais podem ser detectados por todos nós. Quando erros forem detectados, a mensagem é automaticamente anulada e retransmitida até que a operação tenha sucesso.

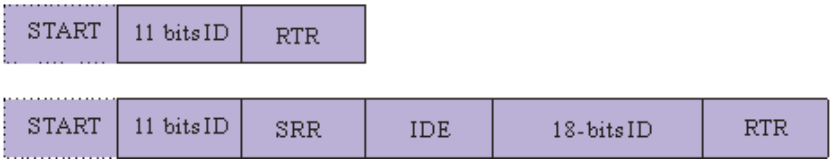
Os quadros CAN são divididos em quatro categorias diferentes: quadro de dados, quadro remoto, quadro de erro e de sobrecarga. Um quadro de dados contém informações da mensagem. Um quadro remoto ocorre quando um nó solicita a outros nós a transmissão do quadro de dados com seu identificador. Quadros de erros é gerado quando um nó detecta algum erro. Os de sobrecarga são gerados quando ocorrem atrasos sucessivos no fornecimento de mais dados ou quadros remotos. Os sete campos da estrutura do quadro dados é mostrado na Figura 16. O campo início (*START*) consiste de um único *bit* dominante e sua borda é utilizada como um gatilho para a sincronização dos restantes *bits* na armação.

Figura 16 – Estrutura do *Frame* de dados CAN

Fonte: Próprio Autor

O campo de arbitragem (ARB) varia dependendo se o quadro possui formato padrão ou formato estendido, como mostrado na Figura 17.

Figura 17 – Campo de Arbitragem (Padrão e Estendido)



Fonte: Próprio Autor

O padrão do campo de arbitragem é utilizar identificador de 11 *bits* que constitui a base para a priorização das mensagens, o menor número ganha e pode transmitir. O quadro estendido utiliza um total de 29 *bits* para o identificador, mas apenas os primeiros 11 *bits* são utilizados para decisões de prioridade. O estado do *bit* RTR (transmissão remota do pedido) é determinado por um tipo de quadro. É dominante em quadro de dados e recessivo em quadro remoto. Devido ao *bit* SRR (pedido remoto substituto) poder ocupar a mesma posição que um RTR em um *frame* padrão, colisões podem ocorrer nos 11 *bits* de identificação entre um quadro com formato padrão e um quadro com formato estendido. Neste caso, o quadro padrão tem a prioridade. A função do *bit* IDE (identificador extensão) varia com o formato do quadro. Em *frames* com formato estendido, ela é usada no campo de arbitragem. No formato padrão, é usada como parte do campo de controle, que inclui alguns *bits* reservados, bem como 4 *bits* que correspondem ao código do comprimento dos dados. O código de comprimento de dados especifica o comprimento da mensagem em *bytes*, usando uma simples contagem binária, a partir de 0000_2 (0 *bytes* de dados) para 1000_2 (8 *bytes* de dados), em que "0" é um *bit* dominante e um "1" é um *bit* recessivo. O campo de dados no padrão pode conter dados de quadros estendidos, onde podem conter de zero a oito *bytes* de informação. O campo CRC contém a sequência CRC e o *bit* recessivo delimitador CRC. A sequência CRC é gerada utilizando todas as sequências de *bits* de quadros anteriores.

A estrutura de um quadro remoto assemelha-se a um quadro de dados sem o campo de dados. É utilizado para solicitar uma transmissão de envio de dados para um receptor. Um quadro de erro contém um *flag* de erro e oito bits delimitadores de erros recessivos. O campo de *flag* de erro é a superposição de todos os *flags* de erro que serão transmitidos. Um erro ativo é indicado por seis consecutivos *bits* dominantes; um *flag* de erro passivo é indicado por seis consecutivos bits recessivos, ao menos que esses bits sejam substituídos por *bits* dominantes de outros nós, transmissão para *flags* de erros ativos. Quadros de sobrecarga funcionam de forma semelhante ao quadros de erro; eles têm um campo de *flag* de sobrecarga que é a sobreposição de todos os *flags* de nós seguido por um delimitador de oito *bits* recessivos. Além dos quatro tipos de quadros, quadro de dados e quadros remotos são separados por um espaço *interframe*.

2.13 CAMADA DE CONTROLE LÓGICO - CAN

As tarefas desempenhadas pela camada de Controle Lógico (LLC - *Logical Link Control*) incluem identificadores de filtros, notificação de sobrecarga, recuperação e gerenciamento. Quando comparado com os protocolos de redes típicas, a característica típica da rede CAN é como os nós são endereçados.

Ao invés de uma fonte específica do endereço de destino, os identificadores estão incluídos no pacote da mensagem. Este campo identificador é processado e encaminhado por todos os nós da rede. Os receptores utilizam filtros pré-configurados para decidir se pretende ou não agir, de acordo com uma mensagem. O filtro lógico realiza uma verificação (comparação) *bit* a *bit* do identificador de entrada no seu próprio banco de filtros. Se ocorrer uma correspondência, a mensagem é processada, caso contrário, a mensagem é ignorada. Se essa situação não é justificada, o sistema permite bits sem importância em um registrador de máscara para ser configurado, o que permite a fiscalização dos nós para recolher as mensagens a partir de muitos identificadores (subordinados), enquanto, ao mesmo tempo, impedem que subordinados se comuniquem uns com os outros. Esse mecanismo é conhecido como o modelo de processamento distribuído "patrão-trabalhador" (HENNESSY; PATTERSON, 2011), que são empregados em diversas áreas como: em automóveis ou ambientes industriais; onde muitos sensores distribuídos se comunicam com um controlador central. O campo de filtros possui um comprimento de 32 *bits* e pode ser configurado com dois, quatro ou oito filtros de aceitação. A escolha da configuração do filtro padrão é baseada na definição de qual formato será utilizado (padrão ou estendido) sobre a organização da rede pretendida.

A configuração de filtros adicionais permite acertos múltiplos, com uma possível configuração de filtros para cada banco. Este recurso é útil para implementar um grupo de hierarquias de comunicação dentro da rede. Devido a diversas possibilidades de configuração, uma terceira parte da aplicação é frequentemente utilizada para definir um conjunto de filtro de parâmetros.

Outra função fundamental da LLC é o *Bit-stuffing*. A fim de manter um número suficiente de transições (para a recuperação do *clock* e para prevenir DC), não mais que cinco *bits* idênticos consecutivos são permitidos. O LLC do transmissor insere automaticamente um bit complementar no domínio após detectar cinco *bits* idênticos consecutivos. Essa ação ocorre apenas nos campos START, ARB, CTRL, DATA e Sequência CRC. O *Bit-stuffing* não ocorre no delimitador CRC, ACK, ou no campo EOF.

A falta de esquemas de gerenciamento de contenção e recuperação são baseados em contadores de erros individuais e o estado de erros de cada nó. Existem cinco tipos de erros que

podem ocorrer durante uma transmissão: um *bit error* ocorre quando um transmissor detecta um *bit* no barramento, que é diferente do *bit* que foi enviado; um *stuff error* ocorre quando seis *bits* consecutivos idênticos são detectados em um campo; um CRC *error* ocorre quando o cálculo do CRC recebido não corresponde à sequência do CRC transmitido; um *form error* ocorre quando um campo contém mais *bits* do que o esperado e um *acknowledgement error* ocorre quando um transmissor não detectar um ACK (resposta) depois de enviar uma mensagem. A validade de uma mensagem é interpretada de maneira diferente para transmissores e receptores. Um transmissor considera uma mensagem livre de erros, se não houver erros até o fim da transmissão do quadro. Para todos os erros que ocorrem, a mensagem é retransmitida automaticamente, com base em regras de prioridade. Um receptor considera uma mensagem livre de erros, se todos, exceto o último *bit* do *frame*, não estiverem corrompidos.

Cada nó no barramento mantém dois contadores de erro - um receptor e um transmissor de erros. Além disso, para cada nó é atribuído um erro de três estados: ativo, passivo, ou barramento desativado. O estado ativo é o estado normal em que um nó ativo envia *flags* de erro (dominante) após a detecção. Nós, no estado passivo, só podem enviar *flags* passivos (recessivo). Um nó em um barramento inativo não pode enviar qualquer *flags* de erro. Os contadores de erro são incrementados e decrementados tomando como base o tipo de erro detectado e o estado de cada nó. O estado de erro de cada nó é determinado pelo valor dos contadores de erro. Geralmente, nós com contadores de erro elevados são colocados no estado passivo ou excluídos do barramento. Se os contadores de erros dos nós são reduzidos para um nível aceitável, eles são autorizados a retornar ao barramento, assumindo o estado ativo.

2.14 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Comentado sobre algumas características das redes industriais e, definida a rede CAN que estará no escopo deste trabalho, vale ressaltar que, poderia ser escolhido qualquer uns dos outros protocolos de redes industriais existentes.

No Capítulo 3 são abordadas as redes *wireless* e definida qual será utilizada no trabalho.

3 PADRÕES DE COMUNICAÇÃO SEM FIO

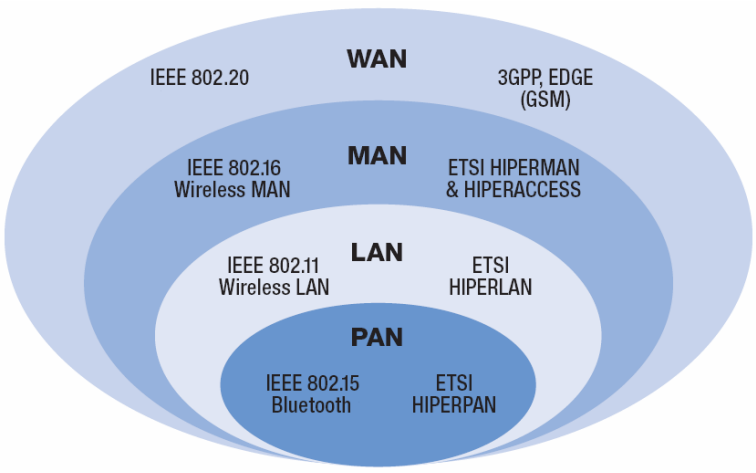
Neste capítulo são definidas as principais tecnologias de comunicação sem fio existentes no mercado. Vale salientar que a análise é elaborada para dar embasamento teórico ao trabalho, indicando assim, que pode ser utilizado qualquer uma das tecnologias para a transferência de dados na comunicação entre os nós.

3.1 INTRODUÇÃO

A procura por tecnologias que buscam interligar pontos de acesso de comunicação se tornou imprescindível nos dias atuais. Assim, é necessário um estudo sobre quais tecnologias estão disponíveis para estas operações de comunicação. De acordo com o IEEE, a comunicação sem fio está subdividida nos seguintes protocolos de comunicação: IEEE 802.11 - LAN sem fio (*Wireless LAN*), IEEE 802.15 - *Wireless Personal Area Network* (*Bluetooth*), IEEE 802.16 - *Broadband Wireless Access* (*WiMAX*) e IEEE 802.20 - *Mobile Broadband Wireless Access* (*MobileFi*).

O IEEE definiu uma hierarquia de padrões complementares para redes sem fio (Figura 18). Essa padronização inclui o IEEE 802.15 para as redes pessoais (*Personal Area Network* - PAN), IEEE 802.11 para as redes locais (*Local Area Network* - LAN), 802.16 para as redes metropolitanas (*Metropolitan Area Network* - MAN) e o IEEE 802.20 para as redes geograficamente distribuídas (*Wide Area Network* - WAN). Cada padrão representa a tecnologia otimizada para mercados e modelos de uso distintos, projetados para complementar os demais. Um bom exemplo é a proliferação de redes locais sem fio domésticas, empresariais e *hotspots* comerciais baseados no padrão IEEE 802.11. Essa proliferação de WLANs está impulsionando a demanda por conectividade de banda larga para a Internet, demanda essa que o padrão 802.16 pode atender oferecendo conexão *outdoor* aos provedores de serviço de comunicação. Para os operadores e provedores de serviço, os sistemas construídos sob o padrão 802.16 representam um terceiro canal (*third pipe*), de fácil implantação, capaz de conectar residências e corporações ao núcleo das redes de telecomunicações em todo o mundo.

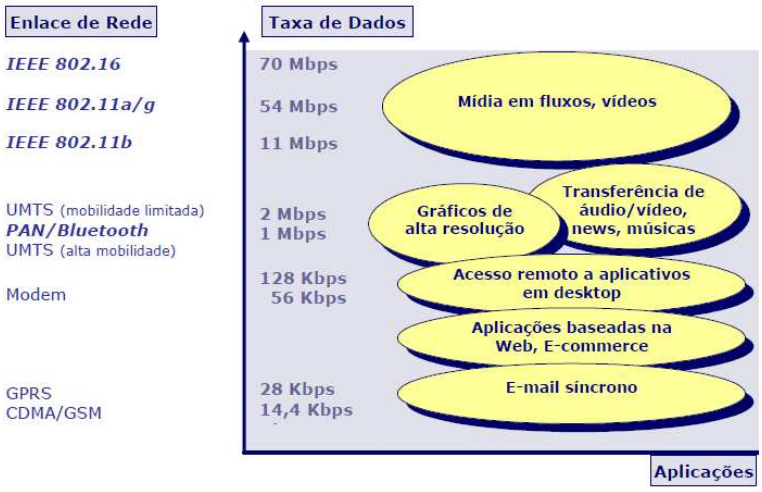
Figura 18 – Padronização global (IEEE e ETSI) para redes sem fio.



Fonte: Adaptado de International Telecommunications Union (2003).

Na Figura 19 ilustra-se diferentes categorias de aplicações, as taxas de dados associadas as mesmas, bem como as tecnologias que melhor se adequam as suas exigências. Dentre as tecnologias ilustradas, estão destacadas aquelas definidas pela família de padrões IEEE 802 para as redes sem fio (Figura 18), representadas na Figura 19 pelos padrões IEEE 802.11, 802.15 (PAN/Bluetooth) e 802.16.

Figura 19 – Cenários de aplicações sem fio.



Fonte: Adaptado de Stepanov e Rosner (2002).

A abrangência geográfica na comunicação entre os dispositivos e a largura de banda de cada uma é que define a escolha do protocolo a ser utilizado. Para este projeto foi escolhido o padrão IEEE 802.15.4 pois opera em pequenas distâncias e possui largura de banda satisfatória para a

aplicação desenvolvida na tese. Vale ressaltar que qualquer outro protocolo pode ser utilizado na comunicação.

A partir deste ponto é definido o padrão escolhido (IEEE 802.15.4) e suas principais características.

3.2 O PADRÃO 802.15.4 - *ZIGBEE*

O atributo fundamental das LR-WPAN é o seu custo mínimo. Os principais fatores que contribuem para o custo mínimo incluem: baixa taxa de dados, o processo de construção da rede e tempo de vida útil da bateria. Na última década, poucas LR-WPANs tiveram sucesso comercial, provavelmente devido a fatores como: custo de fabricação e falta de sensores sem fio baratos. Recentes avanços na fabricação de dispositivos pequenos, tecnologia no desenvolvimento de baterias, e a utilização espectro RF, juntamente com preocupações em termos de segurança têm contribuído para um maior interesse no desenvolvimento e na comercialização destas redes de sensores sem fios. Esta seção apresenta um panorama da mais recente norma IEEE LR-WPAN e fornece um levantamento de algumas aplicações comerciais (KUBAN, 2006).

3.3 CAMADAS DO PADRÃO IEEE 802.15.4 - LR-WPAN

O padrão IEEE 802.15.4 (COMMITTEE et al., 2003) estabelece as bases para as redes sem fio desta investigação e, portanto, é também a base para a discussão do item 3.3.1. A especificação diz respeito apenas às camadas Física (PHY) e Controle de Acesso ao Meio (MAC).

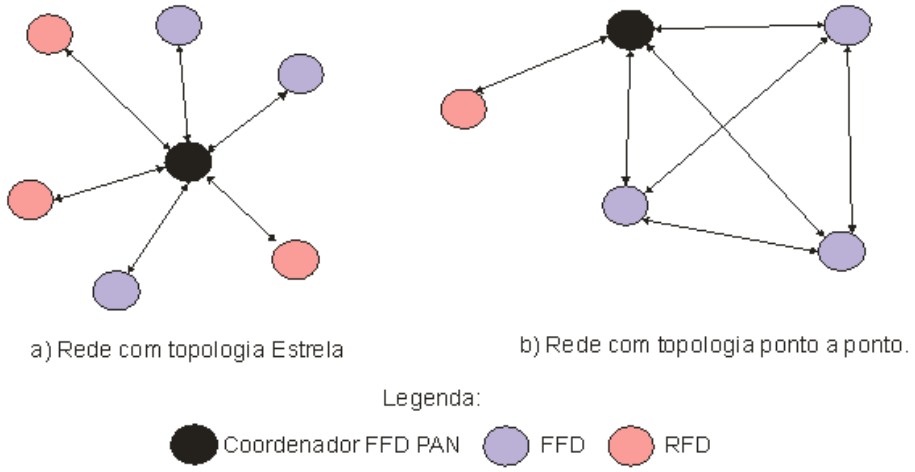
3.3.1 Camada de rede - LR-WPAN

A especificação IEEE 802.15.4 não abrange a Camada de Rede (NWK). As especificações da camada de rede estão atualmente em desenvolvimento pela *Zigbee Alliance* e, no momento, estão disponíveis apenas para os membros dessa organização. Assim, as discussões sobre o NWK camada são limitadas a esta breve panorâmica.

A rede de área pessoal (PAN) deve operar sob a supervisão de um nó coordenador em qualquer topologia de rede permitida, formação em estrela ou ponto a ponto. Conforme mostrado na Figura 20, existem dois tipos de nós que podem se comunicar sobre o LR-WPAN - todas as funções de um dispositivo (FFD - *Full Function Device*) e reduzir as funções de um dispositivo (RFD - *Reduce Function Device*). Um FFD pode funcionar como um coordenador PAN ou como um nó, e comunicar com outros RFDs e FFDs. Um RFD só pode falar com um FFD

e se destina a ser utilizado em um ponto terminal, tal como com um simples sensor ou atuador. O dispositivo adaptável permite a construção de softwares pequenos e menos complexos, resultando em menor consumo de energia e de custos reduzidos nas variantes (KUBAN, 2006).

Figura 20 – Topologias da Rede LR-WPAN



Fonte: Próprio Autor

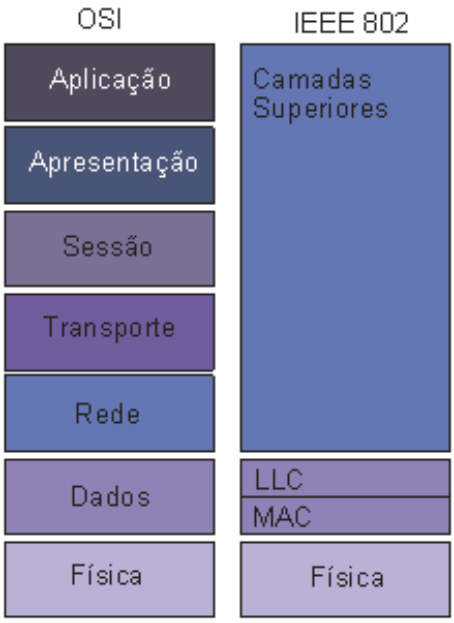
3.3.2 Camada física - LR-WPAN

O sistema opera em relativamente baixas taxas de dados, entre 20 Kbps e 250 Kbps, em várias bandas. Há 16 canais atribuídos em 2450 MHz, 10 canais na banda de 900 MHz e 1 canal em 868 MHz. Os canais são numerados de 0 a 26, com os canais de 11 a 26 reservados para ocupar a banda de 2450 MHz. O tamanho máximo de pacote é limitado a 127 bytes. Um espectro de modulação está empregado, o qual prevê uma resistência mínima de 30 dB (KUBAN, 2006).

3.3.3 Camada de controle de acesso ao meio - LR-WPAN

O sistema utiliza o canal de acesso CSMA-CA e prevê reconhecer plenamente transferência de mensagem para assegurar a confiabilidade. A confiabilidade é uma preocupação crítica na fabricação automatizada e em tempo real para o controle de aplicações. Na Figura 21 (GUTIERREZ et al., 2001) ilustra-se a segmentação da pilha de protocolos do padrão IEEE 802 em relação ao modelo OSI. Um montante considerável da especificação LR-WPAN respeita a Camada Física RF, bem como o estabelecimento de redes *ad-hoc*.

Figura 21 – Modelo OSI e IEEE 802



Fonte: Próprio Autor

Em termos de funcionalidade da interface, existem problemas na camada de dados que afetam diretamente o projeto do mecanismo de extensão.

A subcamada MAC fornece os seguintes serviços:

- *Beacon* Gestão;
- Canal de Acesso;
- *Guaranteed Time Slot* (GTS) *Management* - Gestão de Garantia do tempo de Abertura;
- *Frame* de Validação;
- Confirmação de entrega dos *Frames*;
- Rede de associação e dissociação.

3.3.4 Rede de inicialização e dispositivo de associação

Inicialmente, a rede deve ser configurada pelo Coordenador PAN e os dispositivos devem associar-se com o PAN. Existem vários parâmetros importantes que são inicializados e armazenados no coordenador do PAN. Alguns são definidos em: comprimento do endereço (curto ou

longo), capacidade de segurança e tipo de rede. Uma vez concluído o processo de inicialização, o coordenador PAN entra em modo de espera para receber pedidos de associação.

Em uma rede sem semáforos, um dispositivo que pretenda se associar, primeiro realiza uma busca de detecção de energia no canal. Se o canal estiver ocioso, o dispositivo emite um sinal de solicitação que, simplesmente, verifica se existe um PAN em ação. O coordenador, em seguida, responde com um aviso que incluem muitos parâmetros. Se os parâmetros são compatíveis, o dispositivo envia uma solicitação de associação, que reconhece o coordenador. O dispositivo, em seguida, envia uma solicitação de dados, que é também checada pelo Coordenador. Se o Coordenador aprova, transmite uma associação de resposta, que é reconhecida pelo aparelho. Para um "*token*" de rede ativado, tudo no parágrafo anterior, é aplicado, com exceção do aviso de pedido, que não é necessário porque o coordenador já está enviando avisos periódicos. Uma vez que a associação é confirmada pela resposta do MAC, o dispositivo e o coordenador podem iniciar transferências de dados (KUBAN, 2006).

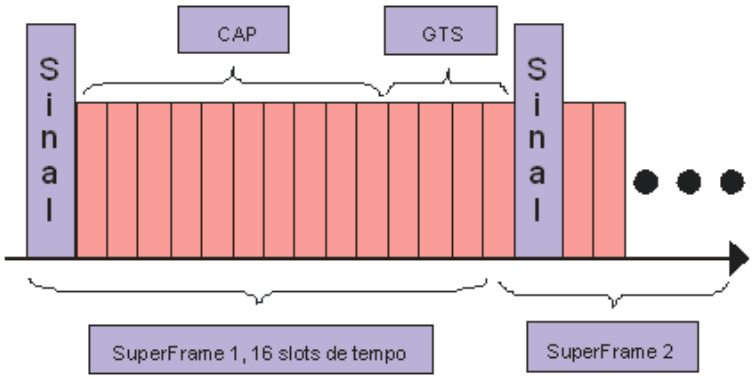
3.3.5 Transferência de dados em redes com sinal ativado

Um dado PAN pode ser configurado como um sinal de rede ativado ou como não-ativado. Em um sinal de rede ativado, *frames* são utilizados para sincronizar dispositivos, identificar o coordenador PAN e estabelecer a necessária superestrutura dentro de uma rede. Uma panorâmica da super estrutura é mostrada na Figura 22. Esta estrutura ilustra a comunicação do arranjo global de muitos nós da rede, com o tempo de conclusão de cada mensagem (e os avisos opcionais associados). Uma vez que o *superframe* está dividido em sinais, o *superframe* inicia com um *beacon*(sinal) e é seguido de 16 intervalos (0-15) de tempos iguais, determinando assim o tempo do *superframe*.

O coordenador PAN determina se o GTS está ativado e, em caso afirmativo, o período livre de contenção é estabelecido, perto do final do envio dos quadros. Se o GTS não estiver habilitado, o período de acesso de contenção (PAC) é de 16 intervalos de tempo iguais após o sinal (*Beacon*) do *slot*. Qualquer dispositivo que deseja se comunicar deve fazê-lo utilizando o método CSMA-CA durante o período de acesso e as operações devem ser concluídas antes da chegada do próximo quadro. GTS é um mecanismo específico para garantir baixa largura de banda e baixa latência nas transferências dos pacotes de dados. Quando ativado, o coordenador PAN dedica até sete faixas de tempo para dispositivos com GTS ativado. Um único GTS pode ser alocado tanto para receber quanto para transmitir, mas apenas um *slot* pode transmitir e um único *slot* pode receber informações por dispositivo. Tecnicamente, para um único dispositivo não são permitidas faixas múltiplas de GTS, mas as faixas de tempo com maior duração podem

ser solicitadas. Por exemplo, um dispositivo pode simplesmente solicitar a transmissão de um único *slot* GTS com comprimento de 7 *slots* regulares, mas isso seria impraticável, pois alocaria toda a largura de banda exclusivamente para o GTS. O coordenador determina se aceita ou rejeita os pedidos GTS, sendo estas informações tratadas pela camada MAC. Se ainda reatarem *slots*, o pedido é aceito, caso contrário, o pedido é rejeitado. Além disso, pedidos GTS pedidos não ocupam o canal eternamente, eles expiram automaticamente após 4 superquadros de não-utilização.

Figura 22 – Estrutura do *SuperFrame* WPAN



Fonte: Próprio Autor

Existem três tipos de transferência de dados suportados:

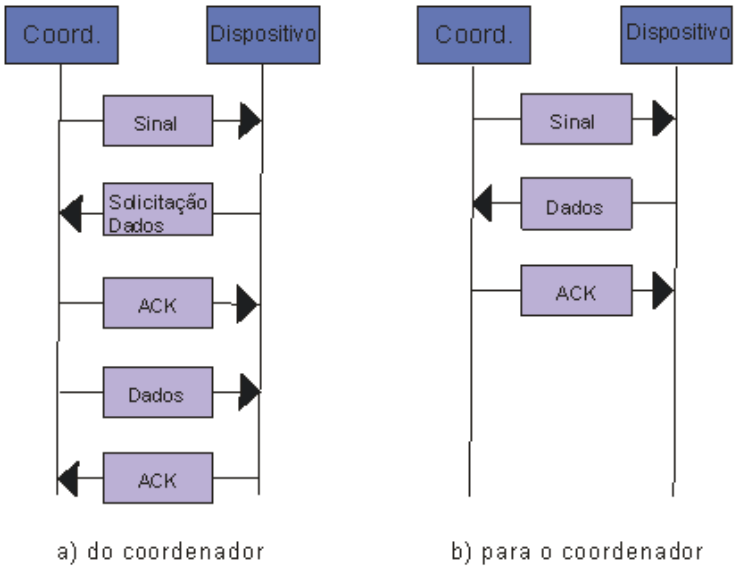
- A partir de um coordenador para um dispositivo PAN.
- A partir de um dispositivo para um coordenador PAN.
- A partir de um dispositivo para outro dispositivo ponto a ponto.

Em uma rede com topologia estrela, apenas os dois primeiros tipos de transferência são utilizados. Estas transações ocorrem de maneira diferente para redes com sinal ativo (*Beacon-Enabled*) e para redes com sinal desativado (*Non-Beacon-Enabled*), apesar de sinais de quadros sempre serem necessários para a associação.

Um sinal de rede ativo é mais útil para situações em que duração da vida útil da bateria e dados periódicos são necessários. Quando o coordenador for configurado como uma rede de sinal ativado, o coordenador FFD envia sinais de transmissão em intervalos regulares. Os dispositivos de funções reduzidas RFDs (alimentação por bateria) podem, então, introduzir um baixa potência (*sleep*) depois de um sinal, e acordar um pouco antes do próximo sinal ocorrer,

assim, conservar o tempo de vida da bateria. Na Figura 23 mostra-se as transações que ocorrem durante uma comunicação entre um coordenador e um dispositivo com sinal de rede ativado.

Figura 23 – Sequência de Transferência de dados com Sinal Ativado



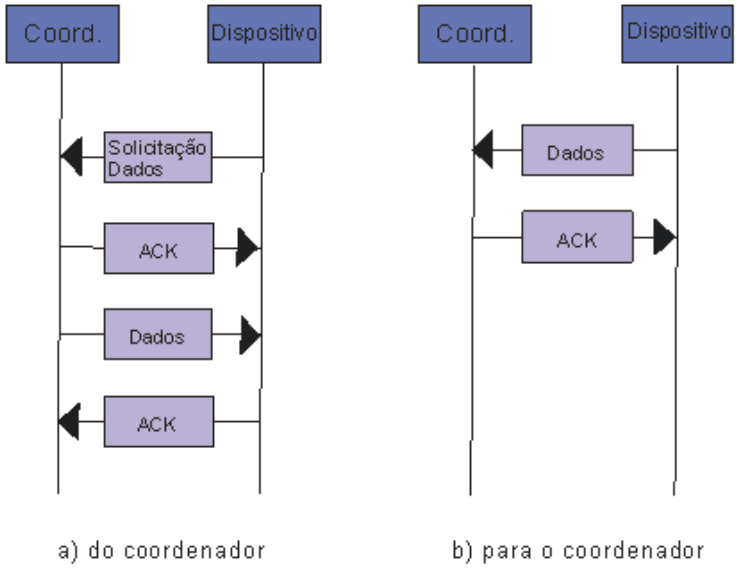
Fonte: Próprio Autor

Como mostrado na Figura 23(a), um coordenador indica com um sinal que uma mensagem está pendente. O dispositivo de escuta detecta o sinal e responde com um pedido de dados. O coordenador opcionalmente reconhece o pedido e, em seguida, envia os quadros de dados. Finalmente, o dispositivo reconhece o recebimento da mensagem. A comunicação com um coordenador é mais rápida, como mostrado na Figura 23(b). O dispositivo simplesmente escuta o sinal e responde com os dados. Após a recepção dos dados, o coordenador dá um aviso que é opcional.

3.3.6 Transferência de dados em redes sem sinal

Redes PANs sem sinal (*Beacon*) funcionam como a maioria das outras redes sem fio. O canal de energia é verificado e, se estiver ocioso, um dispositivo pode transmitir. Se o canal encontra-se ocupado, o dispositivo é desligado e tenta a retransmissão mais tarde. Quando um dispositivo se associa a uma rede, a transferência de dados pode começar como mostra a Figura 24.

Figura 24 – Sequência de Transferência de Dados Sem Sinal



Fonte: Próprio Autor

Uma característica incomum do padrão IEEE 802.15.4 WPAN é que esta rede opera de forma parecida com uma rede cliente-servidor. O Coordenador (servidor) não envia todos os dados para um dispositivo (cliente), a menos que esse dispositivo faça esta solicitação. A principal razão para este regime é consumo de energia - para o coordenador enviar dados diretamente para dispositivos, que deverão estar continuamente ligados à espera de comandos e desperdiçando a vida útil da bateria.

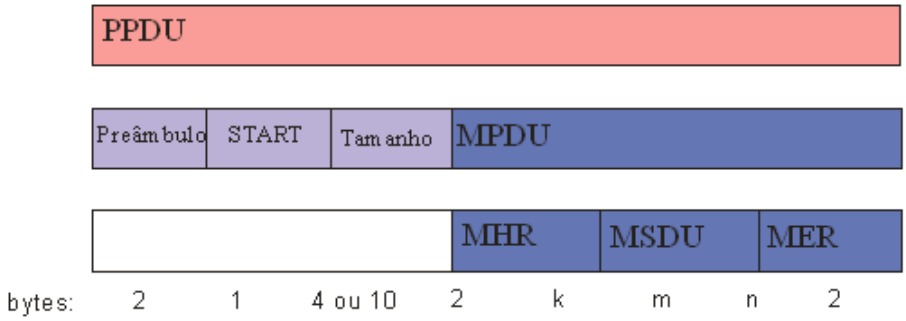
Na Figura 24 ilustram-se duas opções de transmissão de dados - direto e indireto. Com a transferência direta (dispositivo para o Coordenador), o dispositivo simplesmente envia uma mensagem de dados (assumindo que o canal foi inativo), que é recebido, processado e, opcionalmente, reconhecido. Com a transferência indireta (Coordenador de dispositivo), o primeiro dispositivo é responsável pelas sondagens no barramento. Se todos os dados estão prontos para os dispositivos, o coordenador os envia aos destinos.

3.3.7 Organização do *frame*

Dentro de cada transmissão, existem quatro diferentes tipos de estruturas de *frames* possíveis, algumas das quais já foram mencionadas. Sinais de *frames* são utilizadas pelos coordenadores para a sincronização e para a associação da rede. *Frames* de dados levam as informações da mensagem. Os de aviso confirmam o êxito da recepção de uma mensagem. Quadros de comandos MAC (*Medium Access Control*) manipulam entidades de controle de transferências.

A estrutura geral de um *frame* é mostrada na Figura 25. No mais alto nível, o protocolo de dados da unidade física (PPDU - *PHY Service Data Unit*) incorpora todo o quadro. O PPDU pode ser discriminado em um segmento PHY e um segmento MAC, também conhecido como o protocolo MAC de dados da unidade (MPDU - *MAC Control Data Unit*).

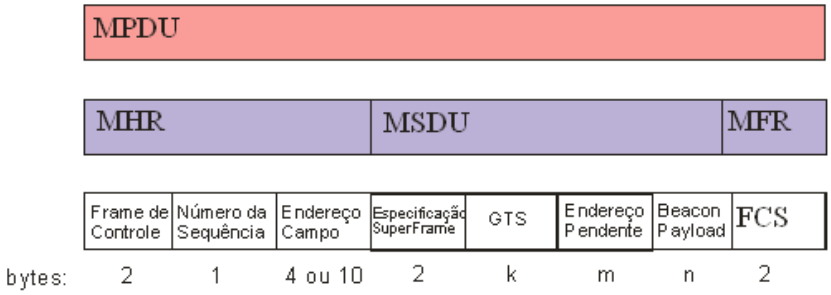
Figura 25 – Sequência de Transferência de Dados Sem Sinal



Fonte: Próprio Autor

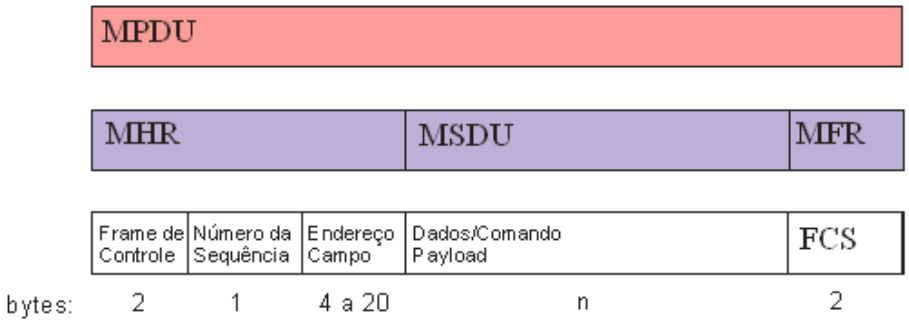
O segmento PHY inclui o campo preâmbulo (4 bytes), delimitador de início (1 byte), e comprimento do *frame* indicador (1 byte). Este segmento é comum a todos os quatro tipos de quadros. O MPDU contém o cabeçalho MAC (MHR - *MAC Header*), o serviço de dados MAC unidade (MSDU - *MAC Service Data Unit*) e do MAC rodapé (MFR - *MAC Footer*). Uma expansão do MPDU é mostrada na Figura 26, onde ilustra uma parte de um farol de moldura MAC. A parte de um quadro de dados MAC e um quadro de comando são mostrados na Figura 27. A única diferença na organização desses quadros é o campo de carga. No entanto, os campos de dados de carga útil são originados em altas camadas e sua transmissão para as subcamadas MAC, enquanto que o campo de comando da carga é gerado dentro do MAC.

Figura 26 – Campos MAC do *Frame Beacon*



Fonte: Próprio Autor

Figura 27 – Campos MAC, *Frames* de Dados e Comando

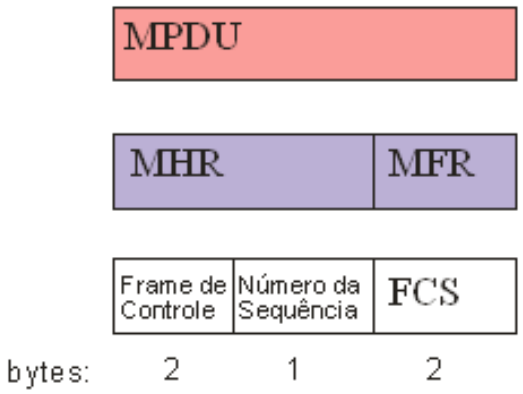


Fonte: Próprio Autor

Os comandos MAC são utilizados pelas entidades de rede para a maioria das atividades, incluindo a associação e dissociação, sinais de notificação, sincronização, horário de gerenciamento, e os pedidos de dados.

Um *frame* de aviso é consideravelmente menor do que o anteriormente mencionado, três quadros, contendo apenas 5 bytes no seu MPDU, como mostrado na Figura 28. Estes quadros são utilizados para confirmar a recepção e validação de um quadro (ou *frame*) de comandos de dados MAC. *Frames* de aviso são opcionais; se for ativado, o transmissor irá reenviar uma mensagem, se o aviso não for recebido dentro do tempo limite. O remetente pode também decidir encerrar a mensagem depois de várias tentativas sem sucesso.

Figura 28 – Campos MAC, *Frames* de Confirmação



Fonte: Próprio Autor

3.4 MECANISMOS DE SEGURANÇA - LR-WPAN

A norma LR-WPAN especifica algumas medidas básicas de segurança na subcamada MAC, incluindo a capacidade de manter uma lista controle de acesso (COMMITTEE et al., 2003) e prevê criptografia simétrica. A aplicação efetiva das funções de segurança, tais como a gestão das chaves e autenticação é deixada para as camadas superiores.

A norma permite quatro serviços de segurança: controle de acesso, criptografia de dados, integridade de quadros e mensagens sequenciais. Sob o controle de acesso, cada dispositivo mantém uma lista de dispositivos com os quais a comunicação é desejada. Quadros de dispositivos indesejados são rejeitados. Criptografia simétrica implica a utilização de uma chave única em ambos os extremos da comunicação. Esta chave pode ser compartilhada por um grupo de dispositivos ou por dois pares que se comunicam dentro de suas respectivas ACLs (*Access Control List*). Uma mensagem de integridade de código (CMI) é utilizada para garantir a integridade do *frame*. A modificação deste código requer o uso da chave criptográfica. O mecanismo é aplicável a *frames* sequenciais ordenados em sequência numérica de que foram transmitidos. O receptor é capaz de determinar se um *frame* é novo ou se teve que ser reenviado, caso os *frames* foram reenviados, estes podem ser rejeitados (KUBAN, 2006).

3.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Depois de definidas as principais redes de comunicação sem fio e também as características do padrão IEEE 802.15.4 que é utilizado no trabalho, no Capítulo 4 são abordados as características da técnica de tabelas de dispersão. Esta técnica foi utilizada para fazer com que a comunicação do protocolo da rede industrial e o padrão IEEE 802.15.4 consumissem menos memória e obtivessem um desempenho maior nas operações de inserção, busca e remoção de dados.

4 TABELAS DE DISPERSÃO - (*HASH TABLES*)

Neste capítulo são abordados os conceitos e a utilização das Tabelas de Dispersão (*Hash Tables*) no gerenciamento de memória em dispositivos embarcados e na melhoria de desempenho para acesso à informação e também a complexidade computacional dos algoritmos utilizados.

4.1 INTRODUÇÃO

O gerenciamento de memória e a busca de melhor desempenho em dispositivos onde o poder computacional é pequeno ou limitado, faz com que a busca de técnica para solucionar tais problemas sejam de interesses primordiais para alavancar a comunicação entre dispositivos interconectados. As redes sem fios, em geral, possuem limitação de memória e processamento em suas operações, sendo assim, conceitos de acesso à informação e gerenciamento de memória através das tabelas de dispersão são imprescindíveis.

O tempo de busca para verificar se uma informação está disponível muitas das vezes pode inviabilizar a construção de métodos para o gerenciamento de memória em pequenos dispositivos como nos microcontroladores. O uso de tabelas de dispersão busca otimizar o tempo de acesso à informação, pois cria índices de acesso direto à informação através das tabelas de dispersão.

Com este objetivo de gerenciamento, otimização e desempenho, abordar-se-á propostas de utilização de tabelas de dispersão em vários trabalhos já publicados que servirão de base para a implementação do trabalho proposto.

Renda, Resta e Santi (2012) utilizaram tabelas de dispersão para abordar o problema de equilibrar o tráfego de dados em uma rede de sensores sem fio. As abordagens existentes permitem balanceamento de carga da rede, alterando o protocolo de gerenciamento usado para encaminhar consultas nas tabelas de dispersão.

No trabalho de Barber et al. (2014) foram desenvolvidas novas técnicas de junção para tabelas de dispersão. Esta técnica consome muito menos memória que as demais e é geralmente mais rápida. A associação *hash* não se restringe às tabelas com encadeamento externo que se encaixam completamente na memória. Este trabalho demonstrou que é possível reduzir a utilização da memória com este modelo, além de se caracterizar por ter um melhor desempenho.

No trabalho de Gao, Groote e Hesselink (2005) é apresentado um algoritmo eficiente para acesso à tabelas de dispersão paralelas com endereçamento aberto, que promete um desempenho mais robusto e confiável do que implementações baseadas em bloqueios convencionais. É garantido que pelo menos um processo irá concluir sua operação dentro de um número limitado de etapas. Para uma única arquitetura de processador a solução é tão eficiente como tabelas *hash* sequenciais. Em uma arquitetura de um multiprocessador, quando todos os processadores têm velocidades parecidas, possuem a mesma eficiência do caso anterior. O algoritmo permite que as tabelas de dispersão possam aumentar ou diminuir, quando necessário.

Em Salami, Arcas-Abella e Sonmez (2015) definiram um projeto que permite armazenar em memória *cache* de forma eficaz, as entradas da tabela *hash* em recursos de blocos de memória de acesso aleatório - BRAM (*Block Random Access Memory*) rápidas, favorecendo, na solução de colisão em hardware. Este projeto permite usar a capacidade total da memória DDR (*Double Data Rate*) para armazenar tabelas de *hash* completas, e empregar um *cache*, para explorar a alta velocidade de acesso de BRAMs.

As crescentes exigências para mapear sequências curtas de DNA de um genoma têm promovido o desenvolvimento de algoritmos e ferramentas rápidas. As ferramentas usadas atualmente são baseados na utilização de uma tabela *hash*. Assim, o método aplicado será uma tecnologia-chave para desenvolver software de mapeamento do genoma mais rápido. Estes métodos podem sem aplicados em plataformas computacionais que possuem restrição de memória e processamento como nos microcontroladores (TAKENAKA; SENO; MATSUDA, 2011. Suplemento).

No trabalho de Tong, Zhou e Prasanna (2015) apresentam que as tabelas *hash* são amplamente utilizadas em muitas aplicações de rede, tais como: classificação de pacotes, classificação de tráfego. O trabalho apresenta uma arquitetura pipeline de alto rendimento de uma tabela de dispersão no FPGA (*Field Programmable Gate Array*). A arquitetura proposta suporta operações de inserção e exclusão para a tabela *hash* que está armazenada na memória. É proposto dois esquemas de acesso na tabela de *hash*: (1) o primeiro esquema atribui a cada entrada de *hash* vários *slots* para reduzir a taxa de colisão das entradas na tabela; cada *slot* pode armazenar a chave de *hash* correspondente à entrada da tabela; (2) o segundo esquema tem uma maior taxa de colisão de entradas, mas uma exigência menor de largura de banda de memória do que o primeiro esquema. O modelo proposto garante que nenhum adiamento é necessário para processar qualquer sequência de operações simultâneas, suportando assim, grande latência de acesso à memória externa. Em um FPGA, a arquitetura proposta atinge 66-85 Gbps, sendo executada em uma tabela *hash* com vários números de entradas, vários tamanhos de chaves e com diferentes valores de latência de acesso à memória DRAM. Também é evidenciado a escalabilidade

em termos de taxa de transferência para configurações de várias entradas de *hash*.

De acordo com Kumar e Crowley (2005), os benefícios de desempenho são significativos com a utilização das tabelas de dispersão, o custo médio de busca é reduzida em 40% ou mais, enquanto que a probabilidade de exigir mais do que uma operação de memória por a busca é reduzida em várias ordens de grandeza.

O trabalho de Xiong et al. (2011) mostra que uma tabela de *Hash* eficiente melhorou os requisitos de desempenho do gerenciamento de registro de conexão em redes de alta velocidade. Com a implementação de uma função *Hash* eficiente e robusta, definindo o identificador de conexão como a sua palavra-chave de entrada, resolveu assim, a colisão de dados a serem manipulados. Os resultados experimentais indicam que com a utilização dos conceitos de tabela de *Hash* apresentou um melhor desempenho em termos de pesquisa e robustez.

4.2 DEFINIÇÃO DE TABELAS DE DISPERSÃO

Nesta seção são conceituadas as tabelas de dispersão e os principais motivos pelos quais foi utilizado nesta tese.

A complexidade de gerenciamento da informação é justamente a tarefa de como encontrar um elemento dentro de uma estrutura de armazenamento, com isso, estruturas de dados e as técnicas de computação para armazenar e localizar uma informação na memória têm foco especial para se conseguir algoritmos mais eficientes. O armazenamento de dados em estruturas ordenadas tornam as buscas mais eficientes. Assim, algoritmos que demandam complexidade de O($log\ n$) são eficientes. Mas com a utilização de tabelas de dispersão (*hash tables*), se bem projetadas, podem ser usadas para buscar um elemento em ordem constante de O(1) (CELES; CERQUEIRA; RANGEL, 2004), ou seja, com a técnica, é possível encontrar diretamente a posição em que o elemento possa estar armazenado.

Devido ao grande número de operações realizadas nas estruturas de armazenamento de dados, e a demora para encontrá-las e processá-las, é necessário realizar isso de forma mais rápida possível (PREISS, 2001). Assim, nesta tese utiliza-se a técnica de tabelas de dispersão para otimizar a busca e diminuir o armazenamento de informações duplicadas em dispositivos onde a memória é limitada, assim, serão definidos os conceitos e técnicas de tabelas de dispersão.

As tabelas de dispersão possuem características para resolver problemas de busca, inserção e remoção de dados.

Ao invés de organizar a tabela de dados segundo o seu valor relativo de cada chave em

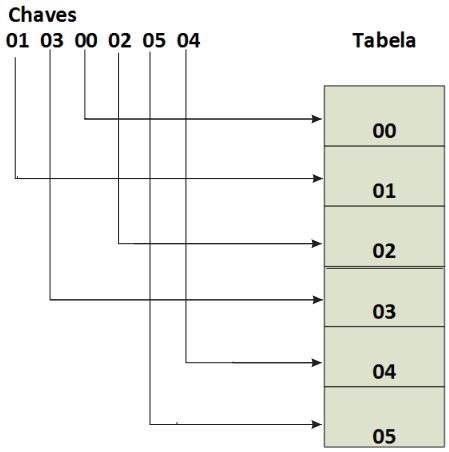
relação às demais, a tabela de dispersão leva em consideração somente seu valor absoluto, interpretado como um valor numérico. Através da aplicação de uma função conveniente, a chave é transformada em um endereço de uma tabela. A intenção é atingir diretamente, se possível, o local onde o elemento se encontra. Na realidade, o método aproveita a possibilidade de acesso randômico à memória para alcançar uma complexidade média por operação de $O(1)$, sendo o pior caso, entretanto, $O(n)$ (SZWARCFITER; MARKENZON, 1994).

4.2.1 Princípio de funcionamento

Suponha que existam n chaves a serem armazenadas em uma tabela T, sequencial e de dimensão m. As posições da tabela se situam no intervalo $[0, m-1]$. Isto é, a tabela é particionada em m compartimentos, cada um correspondendo a um endereço e podendo armazenar r nós distintos. Quando a tabela se encontra na memória, é comum que cada compartimento armazene apenas um nó. Compartimentos com diversos nós são usados geralmente por tabelas de dispersão armazenadas em unidades de disco, quando o número de acessos é mais importante do que o uso eficiente na memória (SZWARCFITER; MARKENZON, 1994). Neste ponto define-se que para melhor desempenho do programa não será utilizado o armazenamento em memória secundária (unidades de disco). O objetivo é armazenar cada chave no bloco referente a seu endereço.

O desenvolvimento das técnicas de tabelas de dispersão foi motivado por um caso bastante simples, porém importante. Suponha que o número de chaves n seja igual ao número de compartimentos m. Além disso, que os valores das chaves sejam, respectivamente, $0, 1, ..., m-1$. Pode-se utilizar então, diretamente, o valor de cada chave como seu índice na tabela. Isto é, cada chave x é armazenada no compartimento z. Esta é uma técnica largamente empregada em programação, conhecida pela denominação de *acesso direto*. A Figura 29 ilustra um exemplo de acesso direto à memória. Os nós são definidos por seu campo chave e as setas indicam os índices das chaves.

Figura 29 – Acesso Direto à Memória.

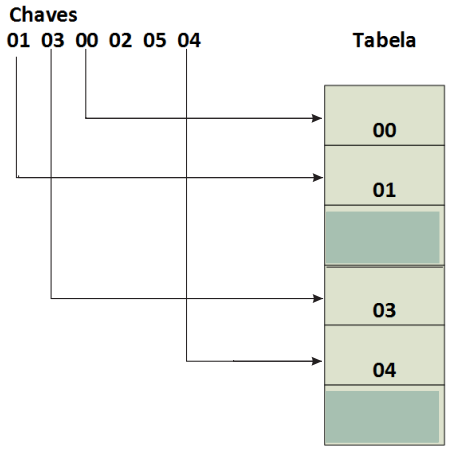


Fonte: Próprio Autor

Um obstáculo aparente é o fato de que as chaves nem sempre são valores numéricos. Essa questão pode ser facilmente contornável, pois a todo dado não numérico corresponde a uma representação numérica no computador. Neste trabalho adotou-se a utilização dos índices a serem armazenados como valores numéricos.

Se a chave ou índice sempre forem valores numéricos, pode ocorrer uma quantidade significativa de espaços vazios na estrutura de armazenamento. Pode-se dar como exemplo um conjunto de duas chaves apenas, de valores de 0 a 999.999, respectivamente. A aplicação da técnica de acesso direto (Figura 30) criaria a uma estrutura de 1.000.000 de posições de armazenamentos, dos quais apenas dois seriam utilizados.

Figura 30 – Acesso Direto à Memória com Espaços Vazios.



Fonte: Próprio Autor

Para resolver este problema deve-se transformar cada chave x em um valor no intervalo de $[0, m-1]$ através da aplicação de uma *função de dispersão h*. Dada a chave x, determina-se o valor $h(x)$, denominado *endereço base* de x. Caso o compartimento $h(x)$ estiver livre, poderá ser armazenado a chave x nesta posição. Infelizmente, a função de dispersão pode não garantir injetividade, pois é possível a existência de outra chave $y \neq x$, tal que $h(y) = h(x)$. Assim, o compartimento $h(x)$ já poderia estar ocupado pela chave y. Este fenômeno é denominado colisão, e as chaves x e y são sinônimas em relação a h (SZWARCFITER; MARKENZON, 1994). Para solucionar este problema é utilizada uma técnica especial denominada de tratamento de colisões. Esta técnica determinará um local alternativo para o armazenamento de x, exceto se a tabela estiver completa.

4.3 FUNÇÕES DE DISPERSÃO

Uma função de dispersão h transforma uma chave x em um endereço base $h(x)$ da tabela *hash*. Uma função de dispersão deve obedecer as seguintes condições (SZWARCFITER; MARKENZON, 1994):

1. produzir um número baixo de colisões;
2. ser facilmente computável;
3. ser uniforme.

A ideia de produzir um número baixo de colisões, indica que uma boa função de dispersão não deve gerar um valor elevado de colisões. Para que não ocorra um número substancialmente grande de colisões, deve ser escolhido quais informações serão utilizadas para gerar os índices e posteriormente definir uma função a partir desta informação que diminua estas colisões.

O segundo aspecto, uma função deve ser facilmente computável pois, o tempo para calcular a função $h(x)$ pode afetar no desempenho da solução. Por exemplo, se for necessária uma informação que não esteja armazenada na memória, e que para isso devemos acessar a informação em disco ou em outros pontos da rede, talvez seja melhor escolher outra função de dispersão que possa ser calculada mais rapidamente.

A uniformidade da função de dispersão, significa que idealmente, a função h deve possuir para acesso a todos os compartimentos de armazenamento de memória, a mesma probabilidade. Isto é, a probabilidade de $h(x)$ seja igual ao endereço base k deve ser $1/m$ para todas as chaves x e todos os endereços $k \in [0, m-1]$ (SZWARCFITER; MARKENZON, 1994).

As funções de dispersão mais utilizadas podem ser definidas por:

1. Método da divisão;
2. Método da dobra;
3. Método da multiplicação;
4. Método da análise dos dígitos.

4.4 MÉTODO DA DIVISÃO

O método da divisão é particularmente fácil e eficiente, sendo por isto muito utilizado. A chave x é dividida pela dimensão da tabela m e o resto da divisão é usado com endereço base. Isto é, $h(x) = x \bmod m$, resultando em endereços no intervalo $[0, m-1]$ (SZWARCFITER; MARKENZON, 1994).

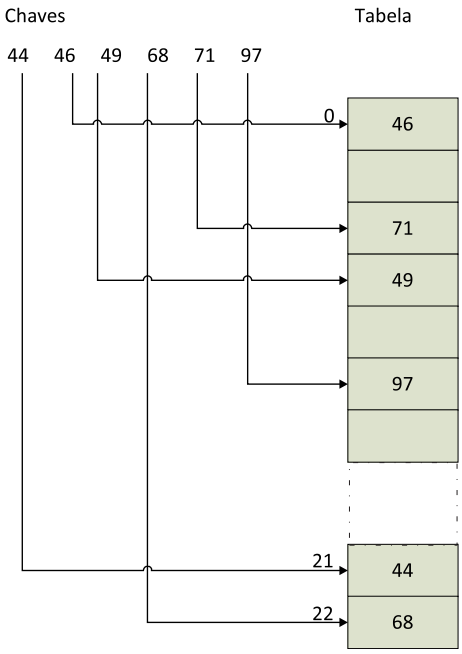
Alguns valores para m são melhores que outros. Por exemplo, se m é um número par, $h(x)$ será par quando x for par e ímpar quando x for ímpar, o que certamente não atende ao problema. Outra situação é ainda pior se m for potência de 2, caso em que $h(x)$ dependerá apenas de alguns dígitos de x. Seja $m = 2^j$ e a chave armazenada numa palavra de memória de 16 *bits*. Se $j = 5$, a função $h(x)$ produzirá endereços que resultam dos últimos 5 *bits* da chave, isto é, $h(x)$ não levará nunca em consideração os dígitos mais significativos de x.

Existem algumas orientações para a escolha de uma boa função de dispersão, tais como:

1. Escolher m de modo que seja um número primo não próximo a uma potência de 2;
2. Escolher m tal que não possua divisores primos menores que 20.

Na Figura 31 apresenta um exemplo de armazenamento de algumas chaves em uma tabela de dispersão de tamanho 23. O resultado utilizando o método da divisão: $44 \ mod \ 23 = 21$, $46 \ mod \ 23 = 0$ e assim sucessivamente (SZWARCFITER; MARKENZON, 1994).

Figura 31 – Função de Dispersão.



Fonte: Próprio Autor

A seção 4.5 aborda possíveis meios de tratamento de colisões em uma mesma posição.

4.5 TRATAMENTO DE COLISÕES POR ENCADEAMENTO

Esta seção é importante porque observou-se que o mesmo endereço base pode ser associado para chaves diferentes, como resultado da função de dispersão, o que dá-se o nome de colisão. Define-se como fator de carga de uma tabela de dispersão o valor $\alpha = n/m$, onde n é o número de chaves armazenadas. Um método para reduzir colisões é diminuir o fator de carga; a medida que este cresce, a possibilidade de ocorrerem colisões também cresce. Esta precaução deve ser tomada, uma vez que o número de colisões cresce rapidamente quando o fator de carga aumenta, mas não resolve o problema. Uma nova chave sempre pode encontrar o seu endereço base já ocupado. Assim, o emprego de tabelas de dispersão implica, necessariamente, a previsão de algum método de tratamento de colisões (SZWARCFITER; MARKENZON, 1994).

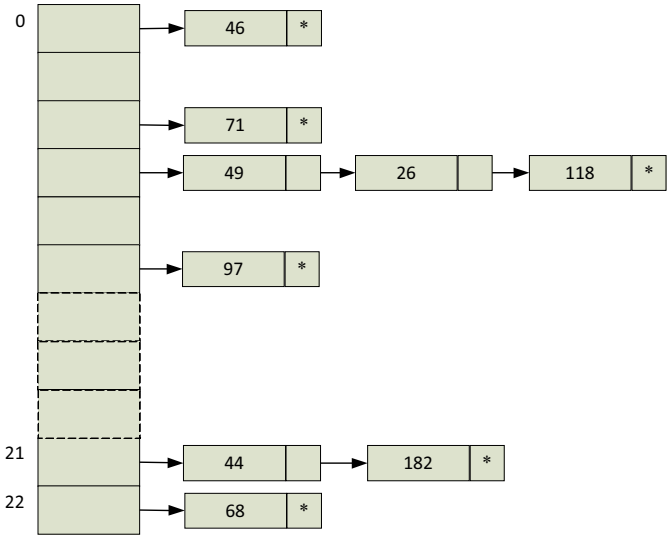
Pode-se tratar as colisões armazenando as chaves duplicadas em listas encadeadas. Existem duas formas de tratamento de colisões, as que podem se encontrar no exterior da tabela ou compartilhar o mesmo espaço da tabela.

4.5.1 Encadeamento exterior

Esta alternativa é muito usada, consiste em manter m listas encadeadas[1], uma para cada colisão geração pela função de dispersão. Na programação, um campo dever ser adicionado em cada nó para o encadeamento. Os nós correspondentes aos endereços base serão apenas nós auxiliares para estas listas. Este método é denominado encadeamento exterior.

Conforme ilustrado na Figura 32, a solução para o mesmo exemplo definido na Figura 31, onde foram acrescentados os elementos 26, 118 e 182 com a mesma função de dispersão $h(x) = x \; mod \; 23$.

Figura 32 – Tratamento de colisão por encadeamento exterior.



Fonte: Próprio Autor

A análise de eficiência desse método para o problema da busca é definido como complexidade de pior caso. Como pode existir mais de uma colisão em um endereço base, estes elementos são inseridos na lista encadeada ligada a estrutura como observado na Figura 32. Como o comprimento da lista pode ser de tamanho n, esta passa a ser a complexidade do algoritmo ($O(n)$), ou seja, complexidade de pior caso, visto que, pode ser necessário percorrer toda a lista para inserir o elemento na lista.

[1] É uma estrutura de dados linear e dinâmica. Ela é composta por células que apontam para o próximo elemento da lista através de estruturas do tipo ponteiro. Para "ter" uma lista ligada/encadeada, basta guardar seu primeiro elemento, e seu último elemento aponta para uma célula nula.

4.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

A estrutura computacional conhecida como Tabelas de Dispersão foi escolhida para ser implementada no gerenciamento dos dados na comunicação entre o protocolo CAN e o padrão IEEE 802.15.4.

No Capítulo 5 estão definidas as ferramentas que deram suporte para o desenvolvimento do trabalho.

5 FERRAMENTAS UTILIZADAS NO DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentados os módulos utilizados para a integração entre o protocolo CAN e o padrão IEEE 802.15.4 - PAN. A integração fará com que os nós da rede sejam capazes de receber dados (pacote IP) através de tecnologia sem fio (IEEE 802.15.4) e capturar informações dos sensores através da Rede de Controle de Área (CAN).

Na seção 5.1 trata de como é realizada a implementação do módulo de rede com diferentes tipos de sensores para monitoramento e a comunicação de uma Rede Pessoal de Área. Em seguida serão apresentados os módulos para a realização do trabalho e suas respectivas características de hardware.

5.1 *KITS* UTILIZADOS PARA A IMPLEMENTAÇÃO E PROGRAMAÇÃO DO MÓDULO

Para o desenvolvimento e integração da rede de protocolos heterogêneos foram utilizados os seguintes módulos para testes e implementações:

- Módulo *STK500* da *Atmel*;
- Módulo de Expansão *STK501* para placas *STK500*;
- Antena de Comunicação *RZ502* para placas de expansão *STK501*;
- *Kit* Arduino UNO R3;
- *XBee Explorer USB*;
- Arduino *XBee Shield*;
- Módulo de Comunicação sem fio *XBee-Pro*;
- Placa de Expansão CAN-*Shield*.

Nos tópicos 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.3.1 e 5.3.2 são apresentadas as características de cada um destes módulos e como foram utilizados para o desenvolvimento do projeto.

5.1.1 Módulo *STK500*

O *STK500* é um *kit* utilizado para desenvolver sistemas para microcontroladores AVR *Flash* da *Atmel Corporation*. Ele foi elaborado para projetar e desenvolver códigos no *AVR STUDIO* além de permitir testes nos novos projetos.

O *STK500* é compatível com o ambiente de desenvolvimento AVR Studio, possui interface RS-232 com o microcomputador para a programação e controle de aplicações, fonte de tensão de alimentação entre 10 V a 15 V *DC*, *sockets* de 8, 20, 28 e 40 pinos para dispositivos AVR, alto nível de tensão para programação paralela e serial de dispositivos AVR, programação serial no sistema (ISP) e reprogramação de dispositivos AVR. Possui 8 *push button*, 8 LEDS acoplados para utilizações diversas, todas as portas de E/S são facilmente acessíveis através dos pinos conectores, uma porta RS-232 adicional e conectores de expansão para módulos auxiliares e área de prototipagem.

Na Figura 33 mostra-se o módulo *STK500* que foi utilizado nos testes de programação e simulação.

Figura 33 – Placa do Módulo *STK500*

Fonte: Próprio Autor

Suporte para novos dispositivos *AVR* podem ser acrescentados de novas versões do *AVR Studio*. Versões mais recentes do AVR podem ser encontradas no site da ATMEL... (2014).

O módulo *STK500* e o ambiente de desenvolvimento *AVR Studio* foram utilizados para testes de implementação e configuração dos sistemas de comunicação.

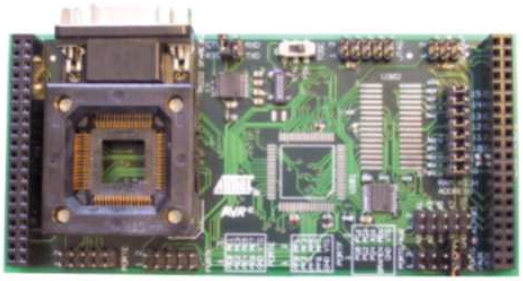
5.1.2 Módulo de expansão STK501

A placa de expansão *STK501* é um módulo elaborado para adicionar os microcontroladores *Atmega*103 e *ATmega*128 à placa de desenvolvimento *STK500* da *Atmel*. Com este *kit*, o *STK500* suporta todos os dispositivos (encapsulamentos) AVR em um único ambiente de desenvolvimento.

O *STK501* inclui conectores, *jumpers* e *hardwares* que permitem a plena utilização das novas funcionalidades do *ATmega*128, enquanto o soquete ZIF (*Zero Insertion Force*) permite a fácil utilização de encapsulamento TQFP para prototipagem. Além de adicionar suporte para novos dispositivos, o *STK501* também fornece suporte para os periféricos anteriormente não suportados pelo *STK500*. Uma entrada adicional $RS-232$ e uma interface SRAM externa.

Na Figura 34 é apresentado o módulo de expansão *STK501* da *Atmel*.

Figura 34 – Módulo de Expansão *STK501*



Fonte: Próprio Autor

O módulo de expansão *STK501* é compatível com o *STK500*. Este módulo foi utilizado devido o *STK500* não possuir suporte para o encapsulamento TQFP disponíveis nos microcontroladores *Atmega*128.

5.1.3 Antena de comunicação RZ502

Para a implementação do protocolo IEEE 802.15.4 foi necessário a integração aos módulos *STK500* e *STK501* a antena de comunicação do sistema. Esta antena está presente no módulo *RZ502* da *Atmel*.

A antena RZ502 é projetada para a avaliação do rádio transceptor (2.4 GHz) AT86RF230 da *Atmel*. Este rádio é totalmente compatível com o padrão IEEE 802.15.4 e pode ser utilizado em tecnologias sem fio de baixa potência, na construção de aplicações residenciais e automação

industrial.

Na Figura 35, é ilustrado o rádio transceptor *RZ502* que pode ser acoplado no *kit* de expansão *STK501* da *Atmel*.

Figura 35 – Rádio Transceptor *RZ502*

Fonte: Próprio Autor

Os módulos *STK500*, *STK501* e a antena *RZ502* foram utilizadas somente para testes dos algoritmos. O valor dos *kits* é um limitador para o desenvolvimento da aplicação. Assim, definiu-se que para implementação da aplicação fosse utilizado módulos *Arduino* com respectivas placas de expansão para o barateamento no desenvolvimento, pois, todos os módulos são compatíveis com microcontroladores *Atmel*.

5.1.4 Arduino UNO R3

O Arduino consiste em uma plataforma de prototipagem em eletrônica - *open source*, elaborado por Massimo Banzi e David Cuartielles em 2005 na Itália, e tem como objetivo facilitar o desenvolvimento de projetos, desde os mais simples aos mais complexos. Com esta plataforma é possível controlar diversos sensores, motores, *leds*, dentre vários outros componentes eletrônicos (ELETROGATE, 2015).

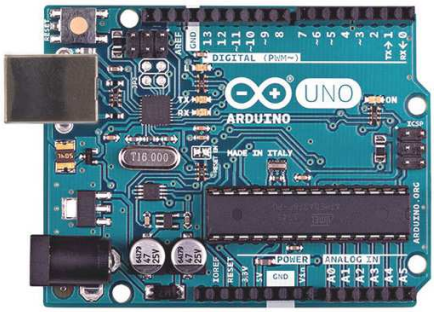
Um ponto forte sobre o Arduino, é que todo material disponibilizado pelo fabricante, como a IDE de desenvolvimento, bibliotecas e até mesmo o projeto eletrônico das placas são *open source*, ou seja, é permitida a utilização e reprodução sem restrição sobre os direitos autorais dos idealizadores do projeto. Porém o nome Arduino, logotipo e o design gráfico de suas placas são registrados e protegidos por direitos autorais. (ELETROGATE, 2015).

5.1.5 Hardware do Arduino

O Arduino da Figura 36 possui algumas características como:

- Memória RAM (utilizada para guardar dados e instruções, volátil);
- Memória flash (utilizada para guardar o *sotware*, não volátil);
- Temporizadores (*timers*);
- Contadores;
- *Clock* do sistema.

Figura 36 – Arduino UNO R3



Fonte: Próprio Autor

Muitos microcontroladores possuem memória reduzida e menor poder de processamento, característica da maioria dos sistemas embarcados. O Arduino Uno R3, por exemplo, possui as seguintes especificações de *hardware*:

- Microcontrolador: *ATmega*328;
- Portas Digitais: 14;
- Portas Analógicas: 6;
- Memória *Flash*: 32 KB (0.5 KB usado no *bootloader*);
- SRAM: 2 KB;

- EEPROM: 1 KB;
- Velocidade do *Clock*: 16 MHz.

5.1.6 Alimentação do Arduino

O circuito interno do Arduino é alimentado com uma tensão contínua de 5 V, isto quando é conectado a uma porta *USB* do computador. Esta conexão fornece a alimentação e também a comunicação de dados. Caso seja necessário é possível utilizar uma fonte de alimentação externa, que forneça uma saída dentre 7.5 V e 12 V contínua, ou pode ser ligada diretamente na placa utilizando os pinos *Vin* e *Gnd* (ELETROGATE, 2015).

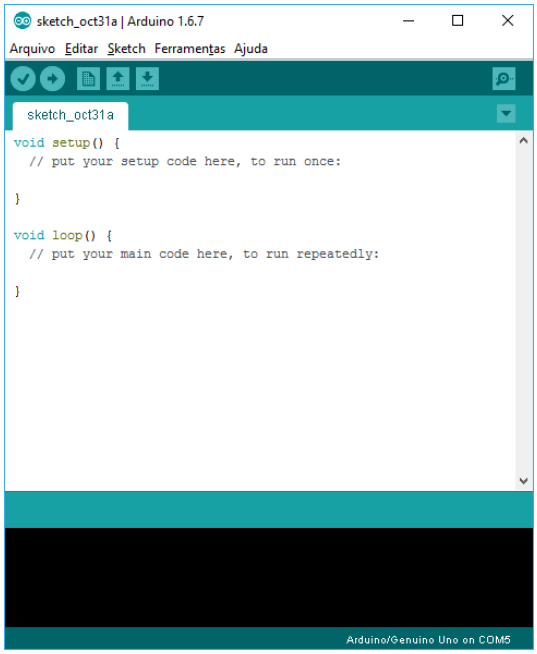
5.1.7 *Software* para implementação de sistemas

O *Software* é utilizado basicamente para escrever o código do programa, salvá-lo, compilá-lo, e realizar a gravação do código no Arduino (memória *flash*) através da porta *usb* do computador. A IDE - *Integrated Development Environment* (ambiente de desenvolvimento integrado) do Arduino foi utilizada para realizar estas tarefas. Este ambiente de desenvolvimento é baseado no *Framework Wiring* e na linguagem de programação C/C++ (ELETROGATE, 2015) sendo compatível com os sistemas operacionais Windows, Linux e Mac OS.

A partir do momento em que se utiliza uma fonte de alimentação externa, o Arduino se torna uma placa totalmente independente. Mas antes de tudo é preciso obter os arquivos de instalação e *drivers*, que vêm juntos no mesmo pacote, que pode ser obtido no site oficial do Arduino.

Na Figura 37 verifica-se a interface da IDE (*Integrated Development Environment*) de programação do Arduino. A linguagem C/C++ é padrão do Arduino.

Figura 37 – IDE de programação do Arduino.



Fonte: Próprio Autor

A IDE do Arduino é muito simples e objetiva, tornando todo o processo de desenvolvimento e gravação bastante intuitivo. Um código desenvolvido para Arduino é chamado de *Sketch*, traduzindo do inglês ao pé da letra seria algo como “esboço” ou “rascunho”. Isso nos dá uma ideia de que nunca terminamos um código, sempre haverá melhorias e novas funcionalidades.

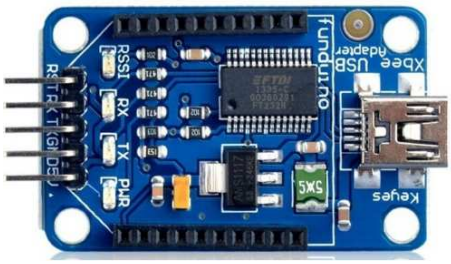
Por tratar-se de um compilador cruzado, o módulo Arduino é conectado ao computador pela interface USB e o sistema operacional (Windows, Linux ou Mac OS) associa uma porta serial à ele (COMx, /dev/ttyUSBx ou /dev/tty-usbserial-x, sendo x a identificação da porta).. Em seguida, após abrir o programa desejado, basta clicar em “*Upload*”. Após executar estes passos, a IDE deverá exibir uma mensagem no final “*Done Uploading*”.

5.2 *XBEE EXPLORER USB ADAPTER*

O *XBee Explorer USB Adapter* é usado para a configuração de parâmetros do módulo *XBee* através do software X-CTU. A comunicação entre a interface e o computador é através de cabo *USB* mini. e com o auxílio do software X-CTU.

O *XBee Explorer USB (Universal Serial Bus) Adapter* pode ser observado na Figura 38.

Figura 38 – *XBee Explore USB Adapter*



Fonte: Próprio Autor

5.3 ARDUINO *XBEE SHIELD*

Para ser possível a utilização do módulo *XBee* em conjunto com o Arduino é possível utilizar uma placa de expansão para que o XBee possa ser acoplado ao sistema. O componente necessário para esta conexão é denominado de *XBee Shield*.

O *Arduino XBee Shield* é compatível com os módulos *XBee* 1 mW, *XBee Pro* 60 mW e *XBee 2.5 Series*. Funciona com *Arduino Uno*, *Duemilanove*, *Mega1280* e 2560.

Na Figura 39 ilustra-se o módulo utilizado para a implementação da aplicação. Nesta placa, acoplamos o móbulo *XBee* para comunicação com outros módulos da rede.

Figura 39 – Placa *Arduino XBee Shield*

Fonte: Próprio Autor

5.3.1 Módulo de comunicação sem fio *XBee-Pro*

O módulo *XBee-Pro* segue o padrão IEEE 802.15.4 e pode ser customizado para diferentes plataformas, funcionando em topologias como *ZigBee/Mesh* e também multiponto. Essa flexibilidade reduz o tempo de *downtime* e facilita o processo de configuração, já que você pode substituir um módulo pelo de outro facilmente.

O módulo *Xbee-Pro* é ideal para aplicações que requerem baixa latência e tempos de comunicação constantes, em redes em configuração ponto-a-ponto e multiponto. Este módulo tem potência de 63 mW, podendo alcançar distâncias de comunicação de até 3200 m.

A configuração do módulo *XBee* pode ser feita facilmente utilizando-se o *software* gratuito X-CTU.

O módulo de comunicação *XBee-Pro* pode ser visualizado na Figura 40.

Figura 40 – Módulo de Comunicação sem Fio *XBee-Pro*

Fonte: Próprio Autor

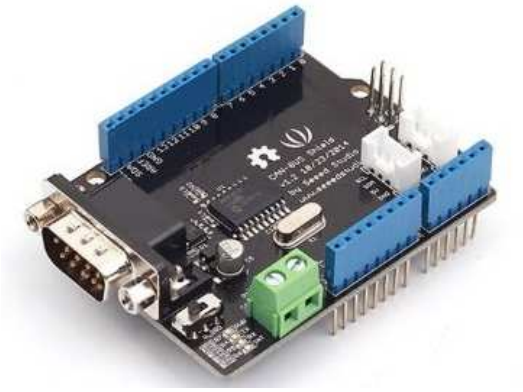
5.3.2 Placa de expansão CAN-*Shield*

Esta placa de expansão *CAN-BUS* possui o controlador CAN MCP2515 com interface SPI (*Serial Peripheral Interface*) e o transceptor MCP2521. As bibliotecas que apoiam o desenvolvimento da comunicação CAN no Arduino disponibilizam todos os materiais pela *internet* gratuitamente.

Atualmente, o *CAN Shield* pode ser anexado as placas *Arduino UNO* ($ATmega328$), *Arduino Mega* ($ATmega1280/2560$) e *Arduino Leonardo* ($ATmega32U4$).

Na Figura 41 mostra-se a placa com suporte ao protocolo CAN.

Figura 41 – Placa *CAN-BUS Shield*



Fonte: Próprio Autor

5.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Definidas as ferramentas utilizadas para o desenvolvimento do projeto e quais metodologias foram utilizadas, no Capítulo 6 é detalhado como foi implementado o módulo de integração entre os protocolos CAN e IEEE 802.15.4.

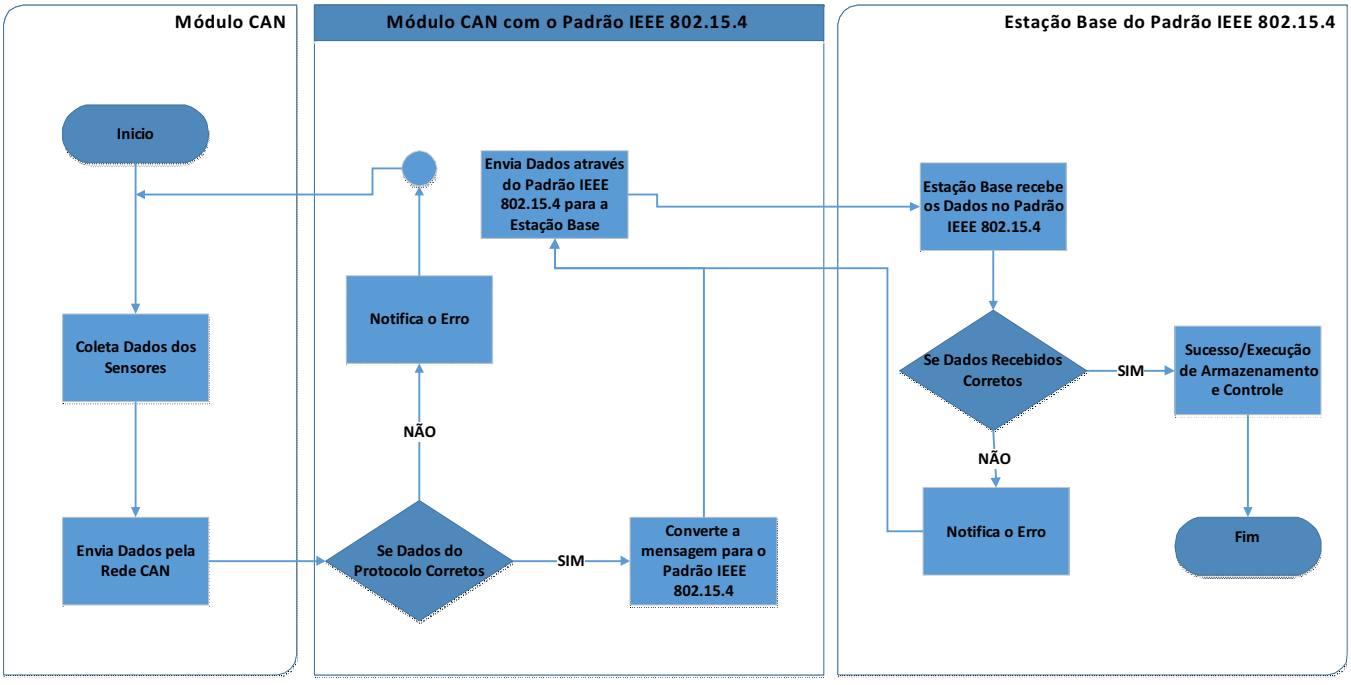
6 INTEGRAÇÃO DA REDE CAN COM O PADRÃO IEEE 802.15.4

Neste capítulo são apresentados os módulos utilizados e as técnicas de programação implementadas para a transmissão de dados entre o padrão CAN e o padrão IEEE 802.15.4.

6.1 DESCRIÇÃO E FLUXOGRAMA DO SOFTWARE IMPLEMENTADO

A integração dos protocolos CAN e IEEE 802.15.4 em um único microcontrolador para a operações de captura e envio de informações, pode ser observada no fluxograma descrito na Figura 42.

Figura 42 – Fluxograma do Sistema



Fonte: Próprio Autor

O fluxograma da Figura 42 é dividida em três módulos:

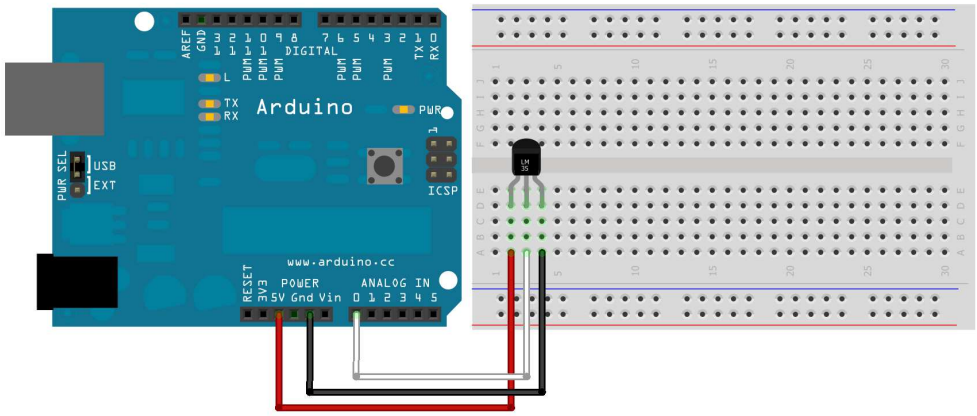
1. Módulo CAN;
2. Módulo CAN com o padrão IEEE 802.15.4;

3. Estação Base do padrão IEEE 802.15.4.

6.1.1 Implementação do módulo CAN

A implementação do Módulo CAN responsável por capturar as informações de um sensor de temperatura (LM35) pode ser especificada da seguinte forma. As informações coletadas são enviadas pelo barramento CAN até ao módulo híbrido (integração do protocolo CAN com o IEEE 802.15.4). O esquemático da conexão do sensor de temperatura ao Módulo Arduino pode ser observado na Figura 43.

Figura 43 – Esquemático Sensor de temperatura LM35



Fonte: Próprio Autor

Como observado na Figura 43 o pino ligado pelo fio de cor vermelha alimenta, o pino de tensão de entrada com 5 V, o pino conectado com o fio preto é o terra do sensor e, por fim, o pino de tensão de saída que se encontra conectado pelo fio branco à porta analógica de valor 0. A tensão lida no pino 0 (zero) é utilizado para efetuar o cálculo de temperatura medido pelo sensor.

Depois de lida a tensão enviada pelo sensor, consegue-se encontrar a temperatura através da seguinte expressão matemática da Equação 1:

$$temperatura = int((float(analogRead(LM35)) * 5/(1023))/0.01); \quad (1)$$

Assim, com o elemento encontrado a ser enviado, este é encaminhado ao módulo híbrido através do barramento CAN.

O circuito com o sensor LM35 em um *protoboard*, conectado ao *CAN Shield* e também ao

módulo Arduino pode ser observado na Figura 45. As coletas realizadas pelos módulos com sensores da rede industrial são enviados para o Módulo de Integração CAN - IEEE 802.15.4 pelo barramento CAN ilustrado na Figura 45. O programa desenvolvido pode ser observado no Apêndice A.1.

6.1.2 Módulo CAN com o padrão IEEE 802.15.4

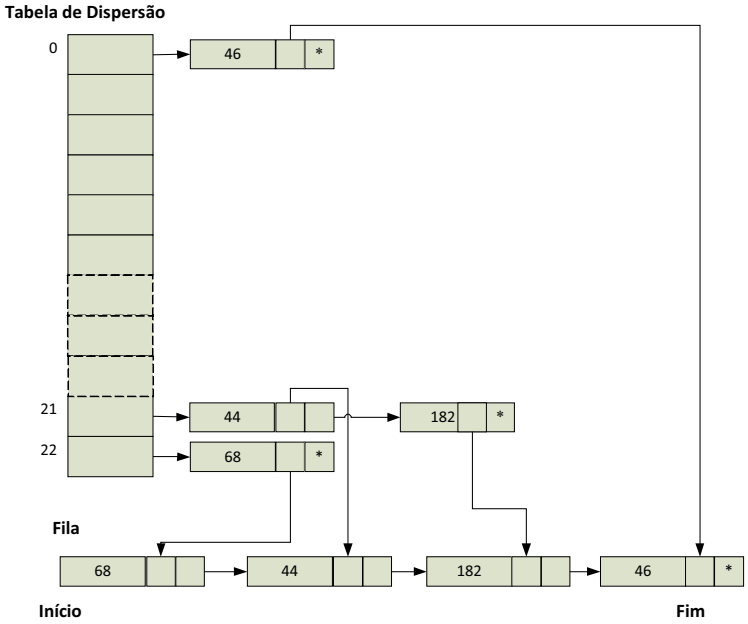
Pode-se dizer que este é o módulo mais importante do sistema, pois converte os pacotes do protocolo CAN para serem enviados pela rede *wireless* seguindo o padrão IEEE 802.15.4. De acordo com o fluxograma da Figura 42, o módulo aguarda os dados serem capturados, caso ocorra algum erro com o pacote enviado pela rede CAN, o erro é notificado e volta-se ao processo de escutar o barramento pelo módulo híbrido. Caso não ocorra nenhum erro com o pacote enviado pelo protocolo CAN ao barramento, o pacote CAN é convertido para o padrão IEEE 802.15.4 e, posteriormente enviados através da rede sem fio. Estas informações podem ser repetidas para outros pontos da rede até que se consiga atingir a estação base do sistema.

O módulo híbrido possui duas características importantes, receber informações de uma rede industrial e posteriormente enviá-las a uma rede *wireless*. Como os dois protocolos possuem taxa de transmissão de dados diferentes, é necessário de que estas informações sejam sincronizadas. Para solucionar este problema, foi proposto a implementação de uma tabela de dispersão que, determina se o elemento a ser armazenado na estrutura já se encontra em sua posição e como a função de dispersão determina em qual posição este elemento será armazenado, fica fácil verificar se o elemento já se encontra na fila para o envio pela rede sem fio. Caso o elemento já se encontre inserido e o último elemento armazenado seja o mesmo que foi inserido anteriormente, a estrutura de controle somente incrementa o valor de quantos elementos deverão ser inseridos pela rede sem fio até que aquele elemento naquele posição esteja esgotado. Caso o elemento seja diferente do último elemento inserido, este é adicionado na última posição da fila de envio, e repete-se o problema da inserção de novos elementos, caso o elemento seja igual ao último elemento inserido, determinado pela função de dispersão da tabela *hash*, um contador incrementa a quantidade de elementos que deverão ser enviados pela rede *wireless*. Após o envio do elemento pelo rede sem fio, o índice que controla a quantidade de elementos que foram enviados é decrementada. Esta técnica pode ser observada na Figura 44 que visualiza como os elementos são armazenados e removidos da estrutura. Através da função de dispersão da Equação 2 definida na Figura 4.5 é observado que a partir do cálculo desta função, encontra-se a posição a ser armazenado do elemento diretamente e posteriormente, o endereço deste elemento é inserido na fila de elementos a serem enviados pela rede sem fio.

$$h(x) = (x) mod (23) \tag{2}$$

Trata-se da tabela de dispersão com endereçamento externo, como explicado no Capítulo 4. No exemplo da Figura 44 o primeiro elemento inserido foi o elemento de valor 68 e armazenado na posição 22 da tabela de dispersão. O segundo elemento a ser inserido na estrutura foi o valor 44 na posição 21. O elemento 182 ocasionou uma colisão na posição 21, sendo necessário o armazenamento em um outro endereço que esteja encadeado/ligado pelo elemento da posição 44 (endereçamento externo - tratamento de colisões). Quando ocorre a colisão e o elemento a ser inserido possui o mesmo valor do armazenado anteriormente, não é necessário que seja alocado um novo espaço de memória, caso ocorra a colisão, uma variável de controle é incrementada, esta variável corresponde a quantos valores devem ser enviados com os dados daquela posição da fila. Assim, se existirem, por exemplo, 5 elementos a serem enviados, o endereço da fila só é desalocado quando todos os elementos forem retirados da fila de envio.

Figura 44 – Funcionamento da Tabela de Dispersão



Fonte: Próprio Autor

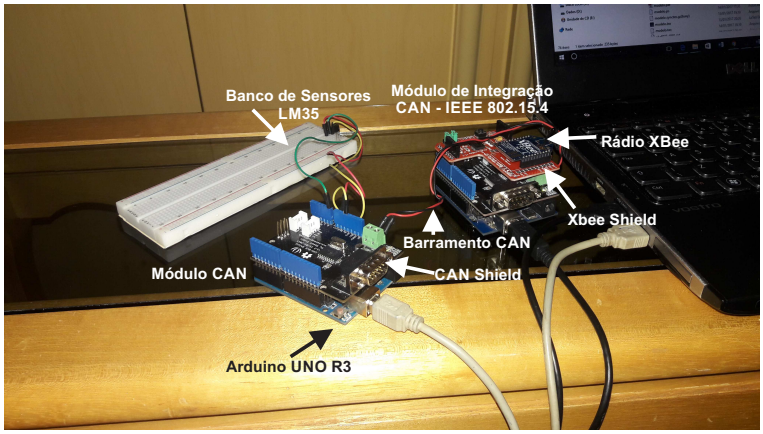
Esta função, utilizando a técnica de dispersão, possui alguns problemas para sincronizar operações de envio e recebimento em protocolos que trabalham de forma diferentes. Para a sincronização do recebimento de uma informação pelo barramento CAN e posteriormente, o envio pelo protocolo da rede *ZigBee*, foi implementado duas *threads*[1] para que estas duas ope-

[1] É uma forma de dividir um processo em várias tarefas e que são executadas em parelelo (concorrente).

rações fossem executadas em paralelo. Assim, foi possível a realização das atividade de envio e recebimento de dados adequadamente.

O conjunto implementado de comunicação e recebimento de dados à partir do Módulo CAN e enviadas para o módulo híbrido podem sem observados na Figura 45.

Figura 45 – Sistema Implementado: Módulo CAN e Modulo de Integração CAN - IEEE 802.15.4



Fonte: Próprio Autor

Pode-se observar na Figura 45 que o Módulo CAN, responsável pela coleta informações sobre temperatura do sensor LM35, está interligado ao Módulo de Integração CAN e IEEE 80.15.4 através do barramento CAN. O Módulo de Integração realiza o envio das informações recebidas através da rede CAN e as envia pela rede sem fio através do Módulo XBee. É importante salientar que as operações estão embutidas em um mesmo microcontrolador (híbrido). O trabalho de Kuban (2006) fazia esta ponte entre os protocolos CAN e o padrão IEEE 802.15.4 mas, utilizava vários microcontroladores para o funcionamento.

Depois de definida a implementação do módulo que gerencia as informações a serem enviadas para os nós roteadores ou para a estação base, foi definido o módulo responsável pela recepção das informações de toda a rede de dados. O código do módulo de integração entre os protocolos CAN e IEEE 802.15.4 pode ser observado no Apêndice A.2.

6.1.3 Estação base do padrão IEEE 802.15.4.

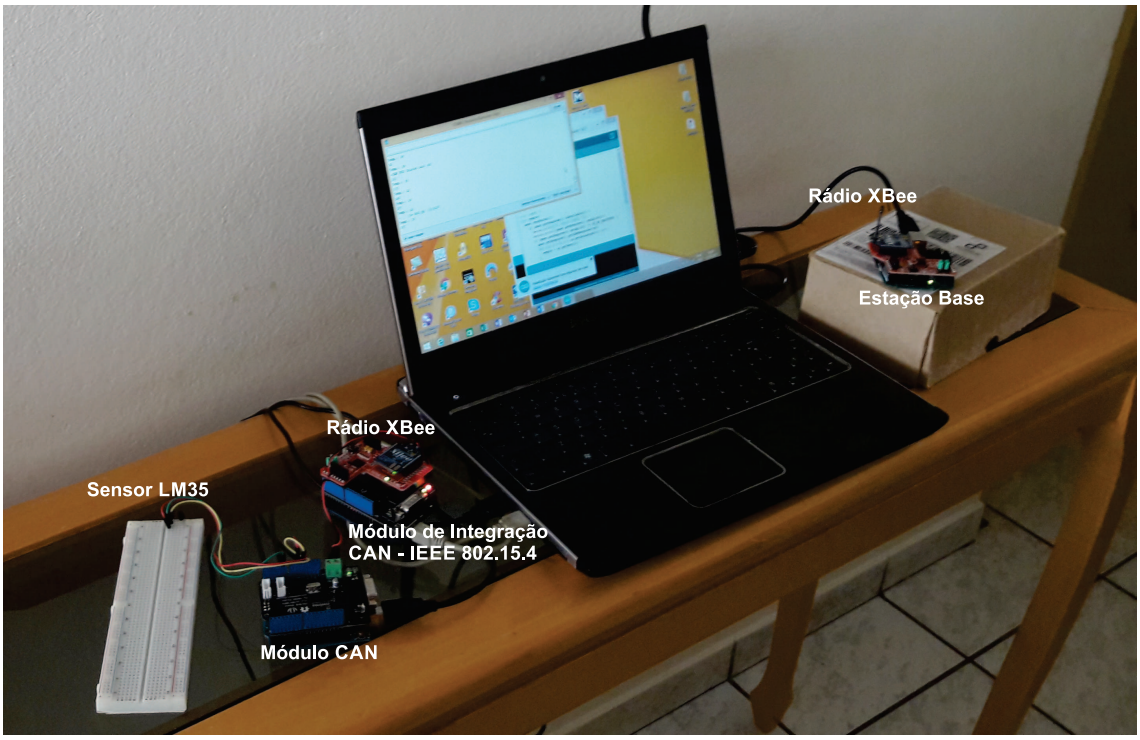
O módulo para o gerenciamento das informações coletadas a partir da rede sem fio podem ser armazenadas em sistemas de banco de dados ou trabalhar com sistemas inteligentes para tomada de decisão na rede de sensores sem fio. Existe a possibilidade de ser implementado sistemas puramente autônomos para o controle da rede de sensores sem fio no que diz respeito

ao controle e monitoramento de sensores e atuadores.

O recebimento das informações com as técnicas e as análises desenvolvidas no Capítulo 7 validam o trabalho e servem como fator estimulante para novas pesquisas. O código utilizado para a estação base pode ser encontrado no Apêndice A.3.

Os Módulos CAN, híbrido e da Estação Base podem ser visualizados na Figura 46. O Módulo da Estação Base coleta as informações enviadas pelo Módulo Híbrido, à partir destas informações, pode-se usar os dados para apoio à tomada de decisões e configurar funções autônomas para o sistema, bastando para isso, que sejam implementadas e programados sistemas computacionais para tratar as informações coletadas/armazenadas.

Figura 46 – Sistema de Comunicação Completo - CAN e IEEE 802.15.4

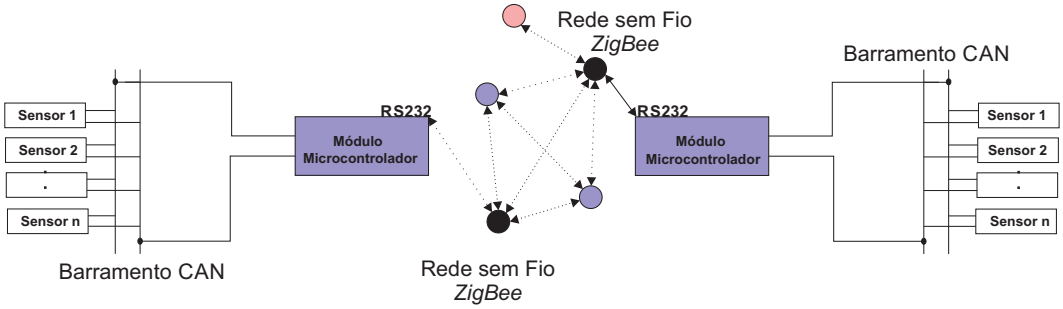


Fonte: Próprio Autor

Através da Figura 46 é definido o diagrama esquemático da Figura 47 que representa o modelo proposto. Na Figura 47 observa-se que os dados colhidos no barramento CAN são enviados para outros pontos da rede utilizando-se da comunicação sem fio do padrão IEEE 802.15.4. Assim, é possível integrar várias redes industriais de lugares diferentes pela rede sem fio. A comunicação da rede sem fio com o barramento CAN também é possível. Esta implementação não foi realizada devido estar fora da proposta de gerenciamento de memória e integração entre os protocolos. Uma vez definido o gerenciamento de memória, a implementação da rede

inteligente com comunicação bidirecional entre os sensores e atuadores é aceitável.

Figura 47 – Modelo do Sistema de Comunicação Completo



Fonte: Próprio Autor

6.2 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram descritos os métodos utilizados para o desenvolvimento e implementação dos módulos para a comunicação do protocolo CAN com o padrão IEEE 802.15.4.

7 RESULTADOS E ANÁLISE DO SISTEMA

Neste capítulo apresenta-se os resultados na implementação do módulo heterogêneo capaz de enviar dados (padrão CAN), processar esses dados, enviar para uma de suas interfaces padronizadas como padrão IEEE 802.1.4 e realizar o monitoramento ou o controle dos transdutores inteligentes interligados ao nó através da estação base.

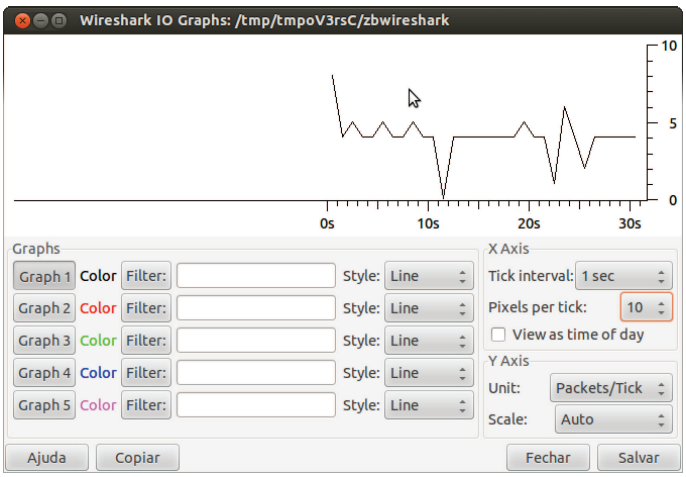
7.1 ANÁLISE DE TEMPO DE RESPOSTA DA REDE SEM FIO - IEEE 802.15.4

Nesta seção é evidenciado as análises do tempo de comunicação entre os módulos da rede através da comunicação sem fio através do padrão IEEE 802.15.4.

A coleta destas informações realizou-se através do software *WireShark* (*software* para análise de protocolos de rede) pertencente ao módulo de análise de rede *KillerBee* (REAVES; MORRIS, 2012). Os dados das análises foram capturados em um cenário onde os módulos se encontravam a uma distância aproximada de 30 metros uns dos outros.

Na Figura 48 mostra-se a quantidade de pacotes enviados na rede sem fio (pacotes/segundo) utilizando o padrão IEEE 802.15.4. Verifica-se que a quantidade de pacotes enviados é em média, de 5 pacotes por segundo.

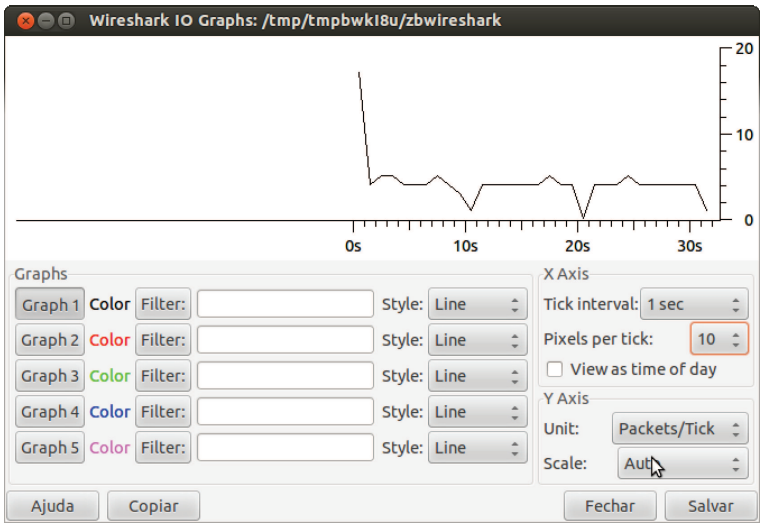
Figura 48 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Quantidade de Pacotes - Análise 1.



Fonte: Próprio Autor

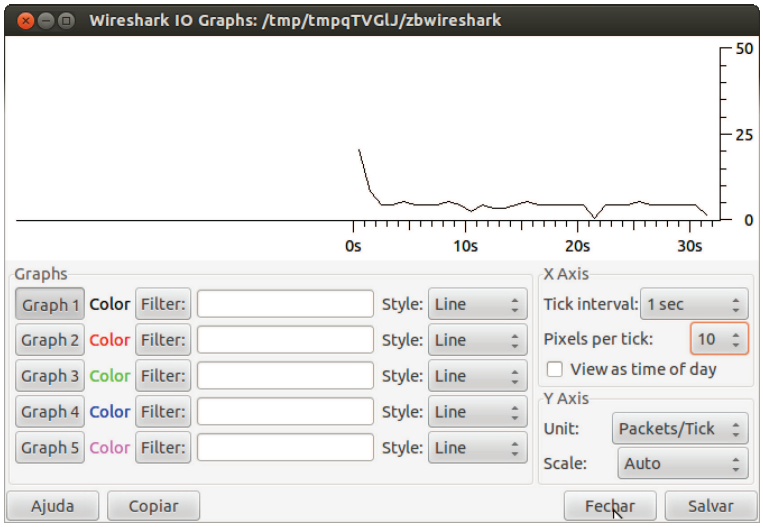
Nas Figuras 49, 50 e 51, apresenta-se as demais análises (Análise 2, 3 e 4) que possuem as mesmas características encontradas na Figura 48, ou seja, enviam em média 5 pacotes por segundo na rede de comunicação.

Figura 49 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Quantidade de Pacotes - Análise 2.



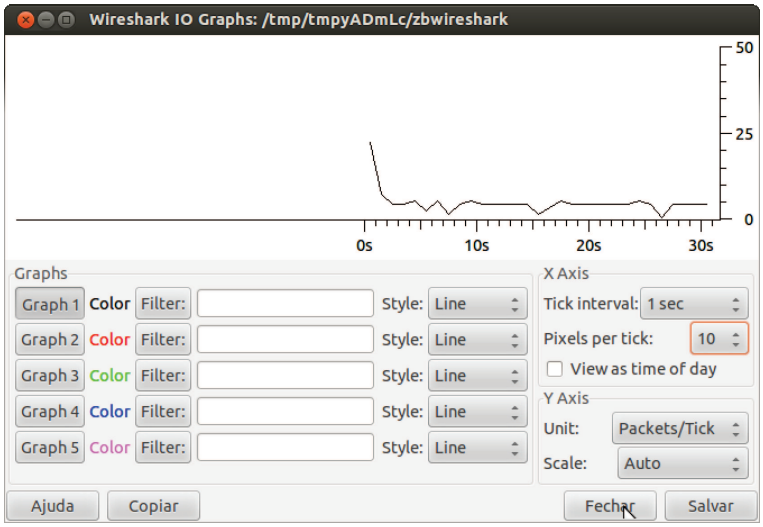
Fonte: Próprio Autor

Figura 50 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Quantidade de Pacotes - Análise 3.



Fonte: Próprio Autor

Figura 51 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Quantidade de Pacotes Análise 4.

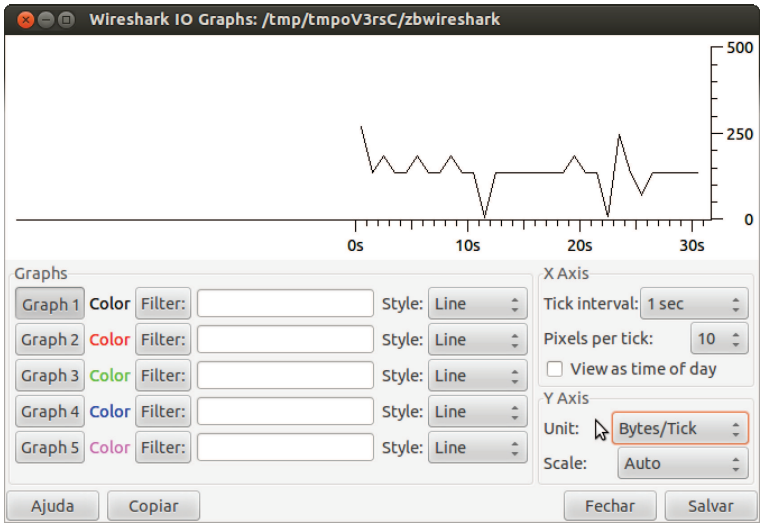


Fonte: Próprio Autor

As análises do tamanho dos pacotes enviados na rede sem fio foram coletas na comunicação entre os módulos XBee através de um *sniffer (RZUSBstick)* de rede para cada simulação realizada. Foram executadas 4 simulações para coletas de informações. Estas simulações estão definidas como Análise 1, Análise 2, Análise 3 e Análise 4.

Na Figura 52 mostra-se o tamanho dos pacotes (Análise 1) que trafegam na rede sem fio (*bytes*/segundo) utilizando o padrão IEEE 802.15.4. Verifica-se que o tamanho dos pacotes enviados é de aproximadamente, em média, de 150 Bps.

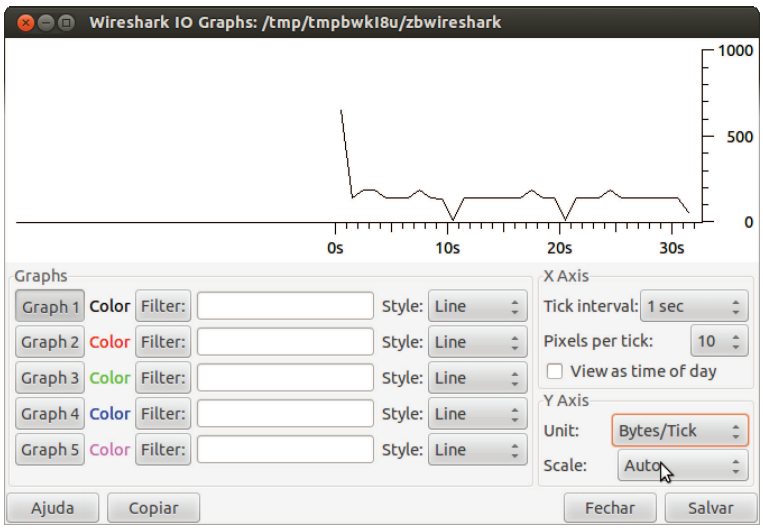
Figura 52 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Tamanho dos Pacotes - Análise 1.



Fonte: Próprio Autor

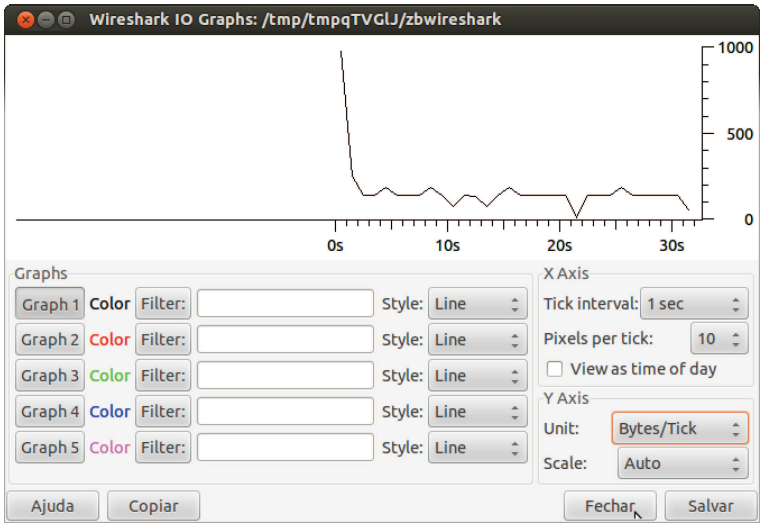
Nas Figuras 53, 54 e 55, encontram-se as análises (Análise 2, 3 e 4), estas análises possuem as mesmas características de encontradas na Figura 48, assim, o tamanho médio dos pacotes enviados na rede é de aproximadamente 150 Bps.

Figura 53 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Tamanho dos Pacotes - Análise 2.



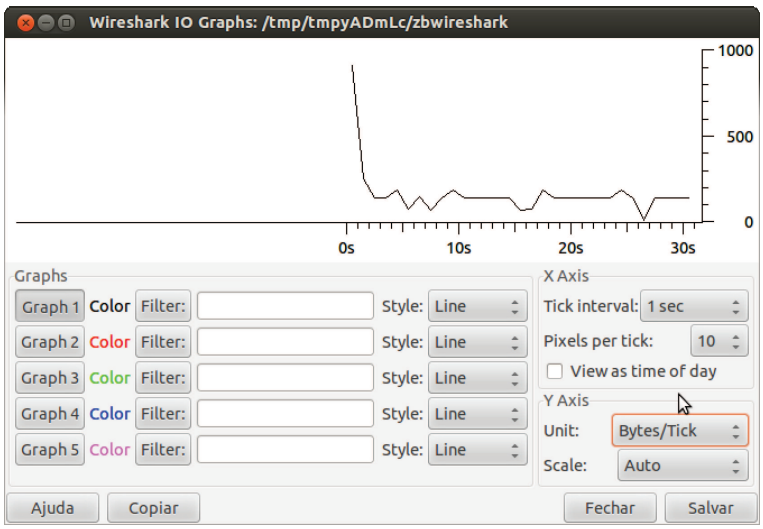
Fonte: Próprio Autor

Figura 54 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Tamanho dos Pacotes - Análise 3.



Fonte: Próprio Autor

Figura 55 – Tempo de Comunicação entre os módulos do Padrão IEEE 802.15.4 - Tamanho dos Pacotes - Análise 4.



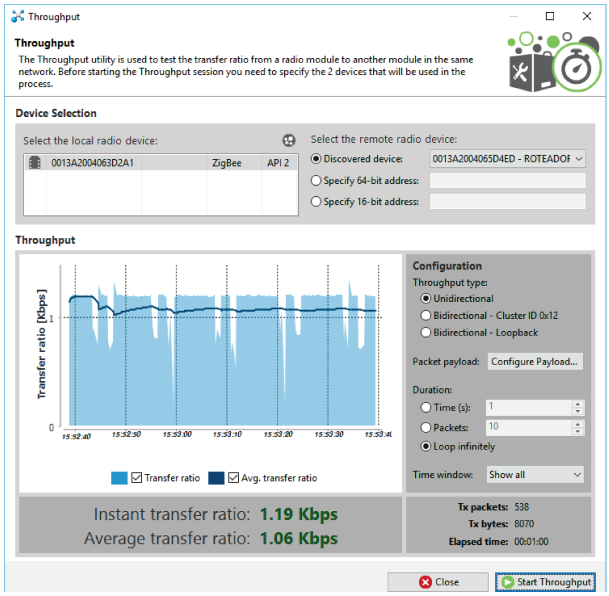
Fonte: Próprio Autor

Para a análise de transferência de dados foi utilizado o software XCTU (2014) que também permite a configuração dos módulos *XBee*.

Na Figura 56 observa-se que a taxa de dados na comunicação entre coordenador e roteador na rede de sensores sem fio através do padrão IEEE 802.15.4 mostrou-se satisfatória. Para uma análise unidirecional (Análise 1), a taxa de dados na transferência instantânea ficou em

1.19 Kbps com uma transferência média de 1.06 Kbps. A quantidade de pacotes analisados foi de 538 durante 1 (um) minuto.

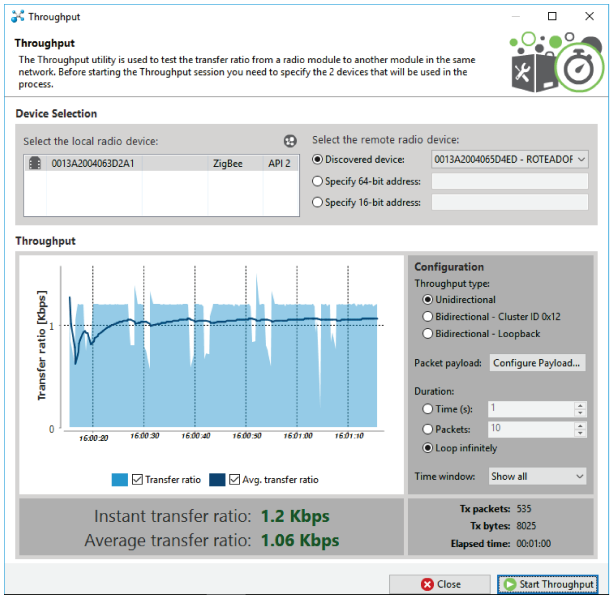
Figura 56 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Unidirecional - Análise 1.



Fonte: Próprio Autor

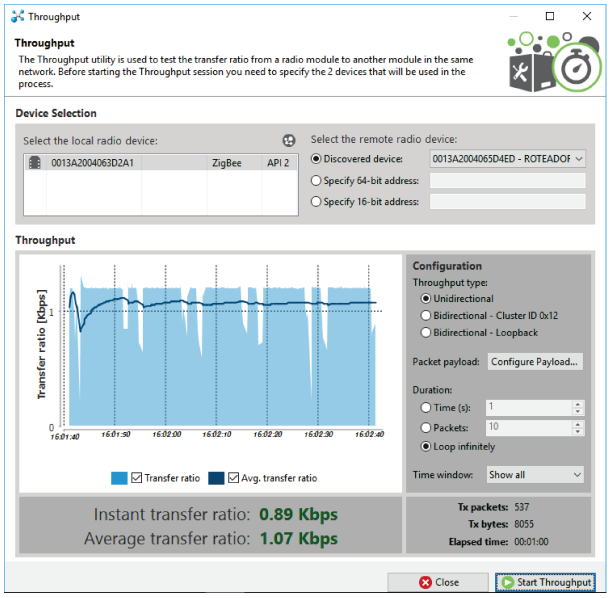
Nas Figuras 57, 58 e 59 é observado que os dados (Análises 2, 3 e 4) possuem as mesmas características que a análise da Figura 56, estas análises obtiveram em média os mesmos valores em média para taxas de transferências.

Figura 57 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Unidirecional - Análise 2.



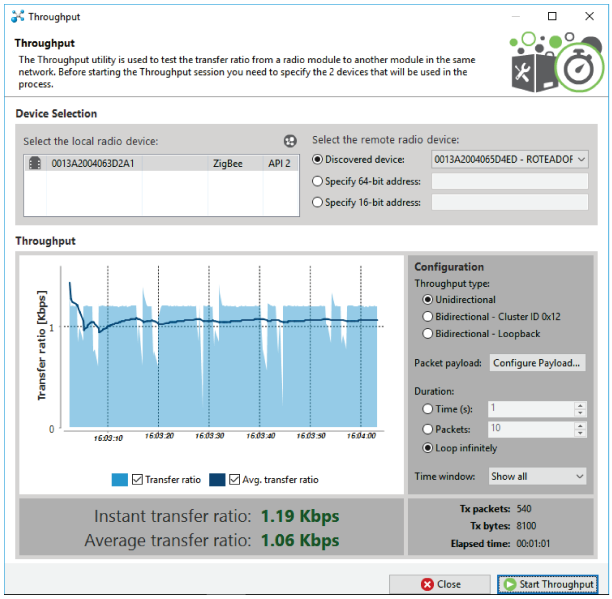
Fonte: Próprio Autor

Figura 58 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Unidirecional - Análise 3.



Fonte: Próprio Autor

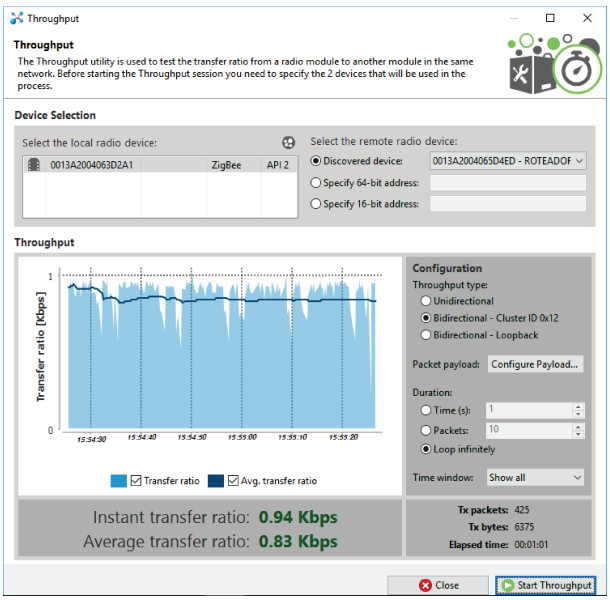
Figura 59 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Unidirecional - Análise 4.



Fonte: Próprio Autor

Para uma análise de comunicação bidirecional, a comunicação entre coordenador e roteador obteve os seguintes valores na transmissão dos pacotes enviados pela rede sem fio. A taxa de dados (Análise 1) na transferência instantânea ficou em 0,94 *Kbps* e uma taxa de transferência média de 0,83 *Kbps* conforme a Figura 60. O número de pacotes analisados foi de 425 durante 1 (um) minuto.

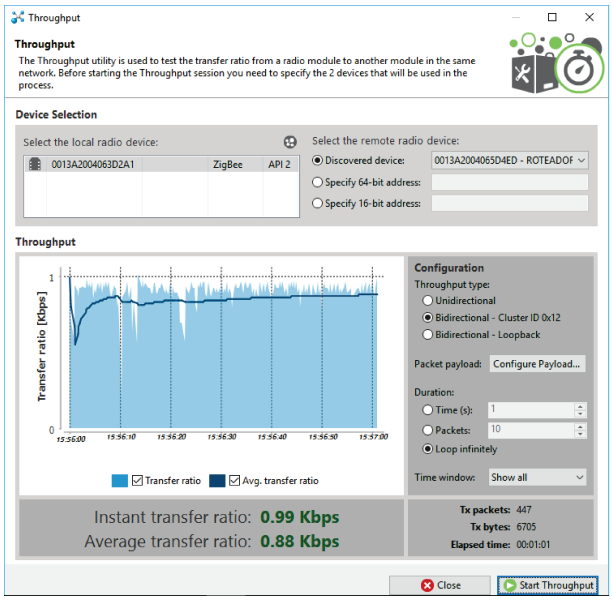
Figura 60 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Bidirecional - Análise 1.



Fonte: Próprio Autor

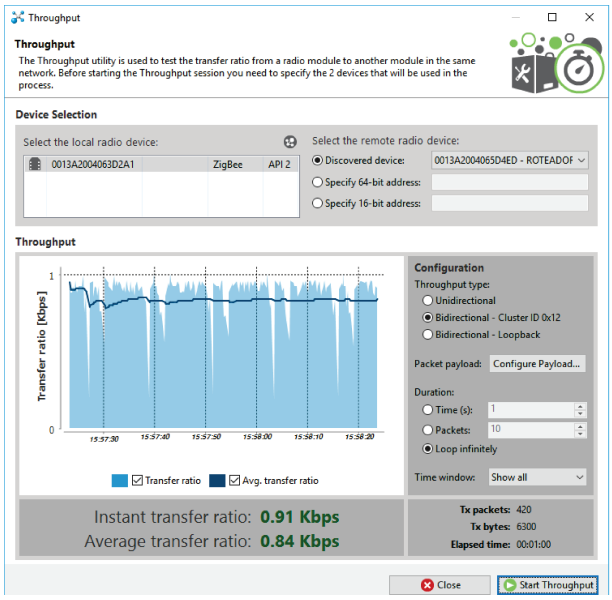
Para melhor evidenciar os dados capturados através do software *XCTU* foram executadas análises que demonstraram possuir as mesmas características da Figura 60. As análises (Análise 2, 3 e 4) das Figuras 61, 62 e 63 comprovam a semelhança da comunicação.

Figura 61 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Bidirecional - Análise 2.



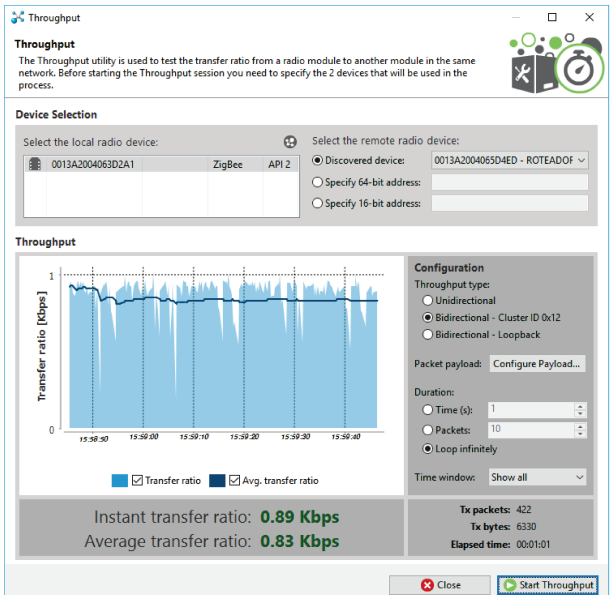
Fonte: Próprio Autor

Figura 62 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Bidirecional - Análise 3.



Fonte: Próprio Autor

Figura 63 – Análise dos Pacotes do Padrão IEEE 802.15.4 - Comunicação Bidirecional - Análise 4.



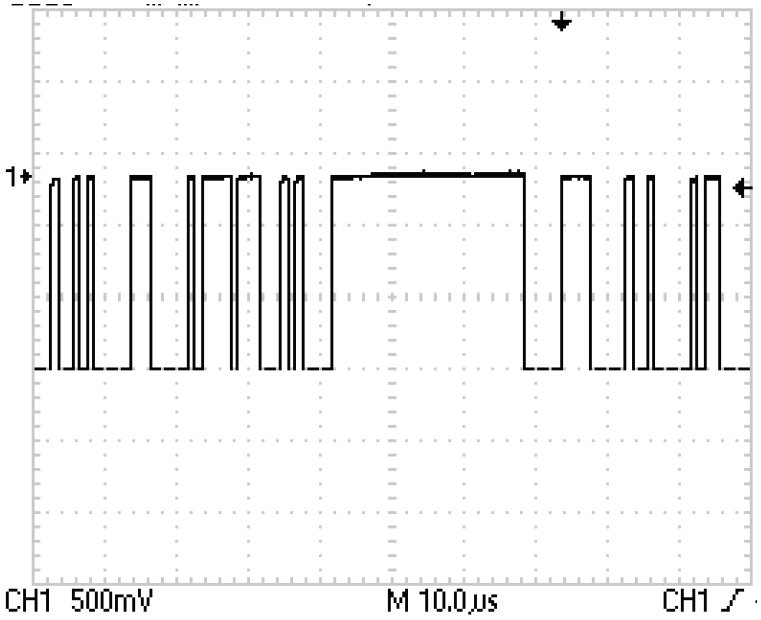
Fonte: Próprio Autor

7.2 ANÁLISE DA TRANSMISSÃO DE PACOTES CAN

Neste seção serão analisados os dados coletados de um nó CAN pertencente a rede construída e que foram enviados para o nó heterogêneo implementado com os protocolo CAN e com o protocolo IEEE 802.15.4.

Na Figura 64 é verificada a forma de onda do protocolo CAN capturada do barramento (Análise 1). Este modelo foi capturado com auxílio de um osciloscópio na operação de envio de dados pela rede. Depois de recebidos pelo nó controlador da rede estas informações são empacotadas no padrão IEEE 802.15.4 e transmitidos para outros pontos da rede através de comunicação sem fio.

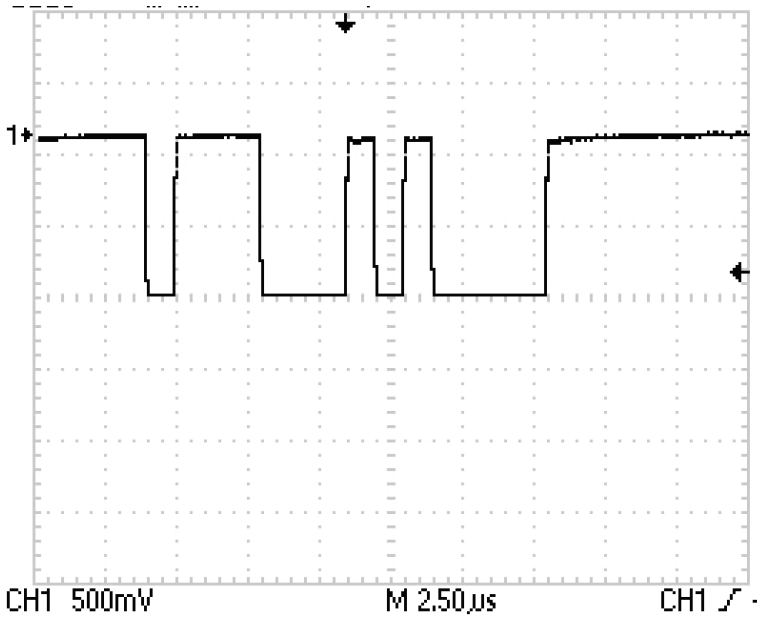
Figura 64 – Forma de onda do Protocolo CAN enviado para o Módulo de Integração - Análise 1.



Fonte: Próprio Autor

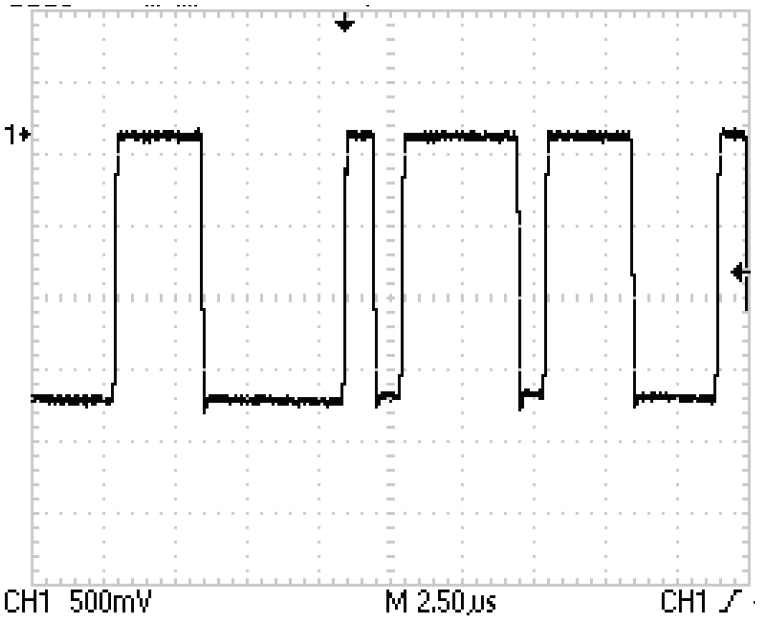
As análises (Análise 2, 3 e 4) das Figuras 65, 66 e 67 possuem a mesma característica da forma de onda da Figura 64 que foram capturadas do barramento CAN.

Figura 65 – Forma de onda do Protocolo CAN enviado para o Módulo de Integração - Análise 2.



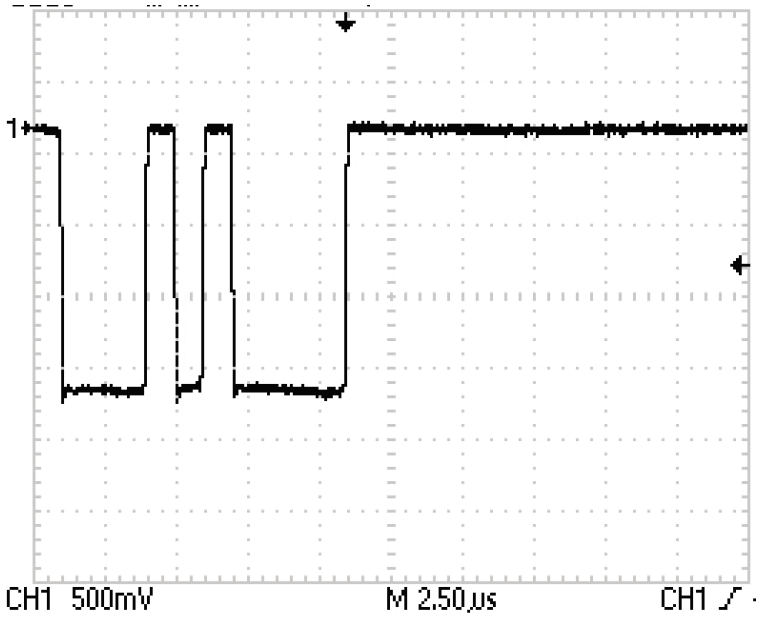
Fonte: Próprio Autor

Figura 66 – Forma de onda do Protocolo CAN enviado para o Módulo de Integração - Análise 3.



Fonte: Próprio Autor

Figura 67 – Forma de onda do Protocolo CAN enviado para o Módulo de Integração - Análise 4.



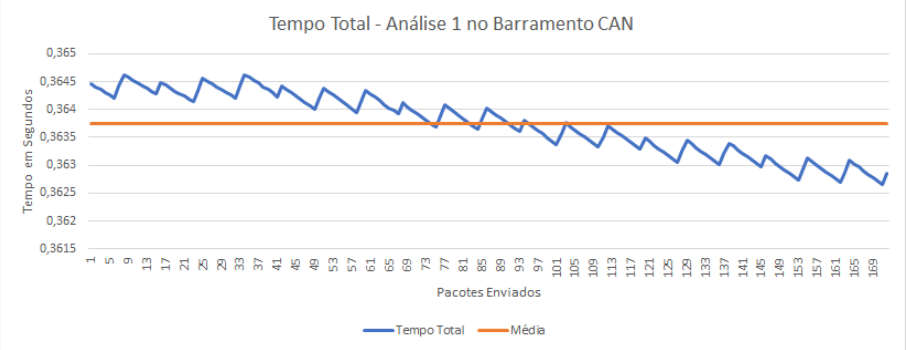
Fonte: Próprio Autor

Em relação ao tempo médio de comunicação entre os nós da rede CAN, estes sofreram uma variação devido a dificuldade de sincronização dos *clocks* dos microcontroladores pertencentes a rede. Para esta análise foi capturado o tempo de envio inicial no módulo CAN e o tempo final

de comunicação no nó CAN que recebeu as informações da rede. Como a sincronização dos relógios não determinam com precisão o tempo de comunicação foi realizado uma estimativa de tempo de comunicação através dos dados capturados.

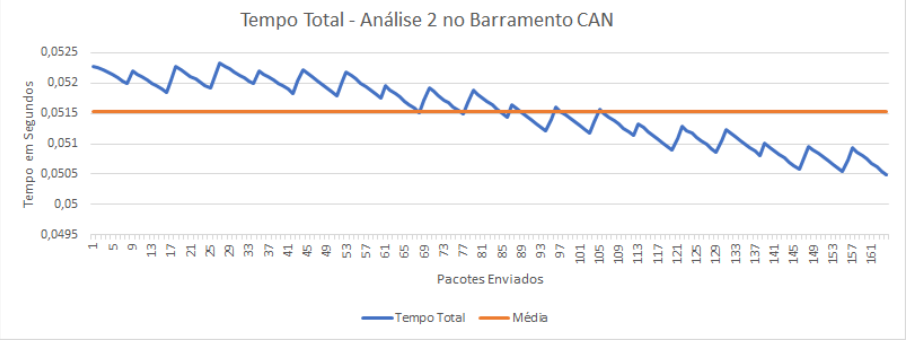
Nas análises (Análise 1, 2 e 3) das Figuras 68, 69, 70 mostram-se uma diferença no tempo de comunicação devido o tempo dos relógios de todos os microcontroladores do sistema não estarem sincronizados, entretanto, nota-se que a característica do gráfico para todos os casos são iguais. Nas Figuras 68, 69, 70 é verificado a quantidade de pacotes enviados na rede e o tempo médio em segundos de cada pacote. Na Figura 68 o tempo médio foi de 0,36 segundos, o da Figura 69 foi de 0,05 segundos e, para a Figura 70 o tempo médio foi de 0,74 segundos. Houve esta variação no tempo de comunicação do CAN devido os relógios (*clocks*) dos microcontroladores não estarem sincronizados.

Figura 68 – Tempo de Envio do Pacote CAN para o Módulo de Integração - Análise 1.



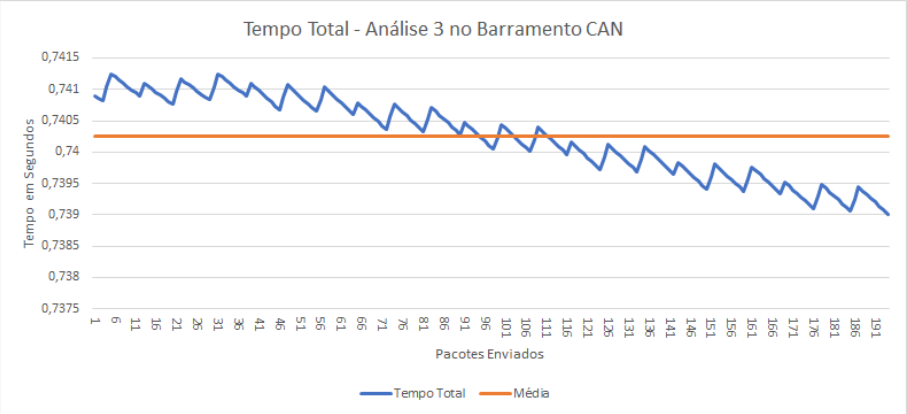
Fonte: Próprio Autor

Figura 69 – Tempo de Envio do Pacote CAN para o Módulo de Integração - Análise 2.



Fonte: Próprio Autor

Figura 70 – Tempo de Envio do Pacote CAN para o Módulo de Integração - Análise 3.



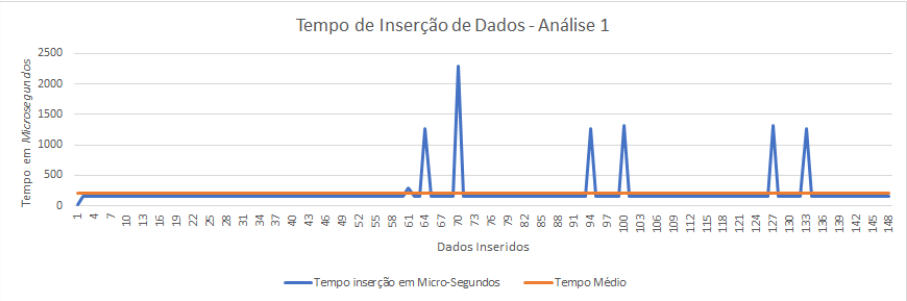
Fonte: Próprio Autor

7.3 ANÁLISE DO GERENCIAMENTO DE MEMÓRIA COM A UTILIZAÇÃO DE TABELAS DE DISPERSÃO

Como a velocidade de localização para saber se um elemento se encontra ou não na tabela de dispersão é direta, através da utilização da função de dispersão da Equação 3 utilizada para os testes no sistema foram coletados o tempo de latência para a inserção das informações na variável de controle. Nas análises (Análise 1 e 2) das Figuras 71 e 72 respectivamente, observa-se o tempo médio de inserção para cada elemento é de 200μs, mas que na maioria das inserções de dados ficaram em torno de 150μs. Isto ocorreu devido a alguns picos na execução da tarefa.

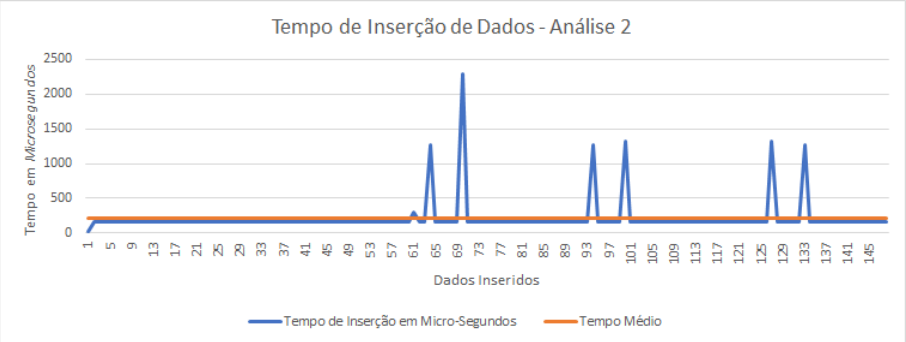
$$f(x) = (x)mod(47) \tag{3}$$

Figura 71 – Tempo para a Inserção de Dados na Tabela de Dispersão - Análise 1.



Fonte: Próprio Autor

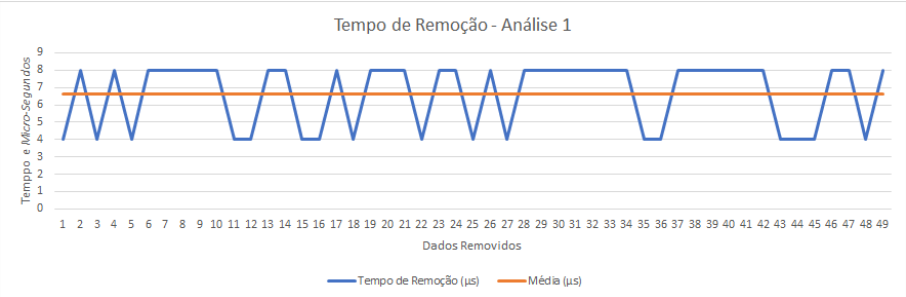
Figura 72 – Tempo para a Inserção de Dados na Tabela de Dispersão - Análise 2.



Fonte: Próprio Autor

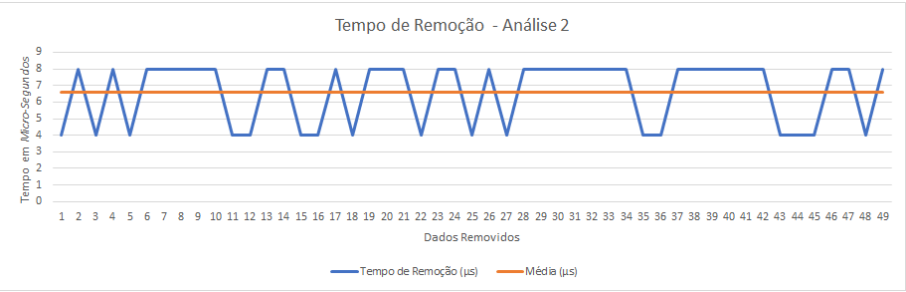
Para a remoção dos elementos da Fila, como é conhecido o início da estrutura, basta enviar os elemento pela rede sem fio e posteriormente removê-lo da memória ou simplesmente decrementar a variável que controla a quantidade de elementos que devem ser enviados. AS análises (Análise 1 e 2) ds Figuras 73 e 74 informam que o tempo médio de remoção de um elemento é de aproximadamente 6.5μs.

Figura 73 – Tempo para Remoção de Elementos da Fila de Dados - Análise 1.



Fonte: Próprio Autor

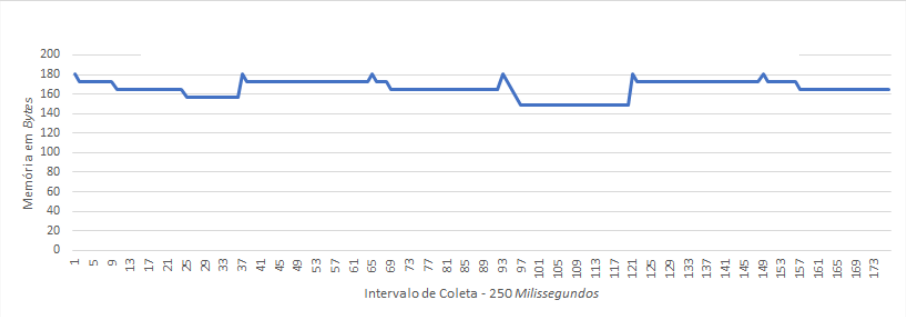
Figura 74 – Tempo para Remoção de Elementos da Fila de Dados - Análise 2.



Fonte: Próprio Autor

A análise do gerenciamento de memória utilizando tabelas de dispersão pode ser observado na Figura 75 - Análise 1. Esta análise foi realizada com a captura de quanto estava sendo utilizado de memória no momento da execução da aplicação. Os dados da Figura 75 foram capturados à partir de um sensor de temperatura pelo módulo CAN e enviado para o módulo híbrido (protocolo CAN e IEEE 802.15.4). O intervalo de tempo para a captura dos valores de memória consumida pela aplicação foi de 250 millisegundos. Coma a varição de temperatura é muito pequena, nota-se que os pontos de linearização no consumo de memória da Figura 75 indica que o algoritmo proposto funciona muito bem para problemas desta natureza.

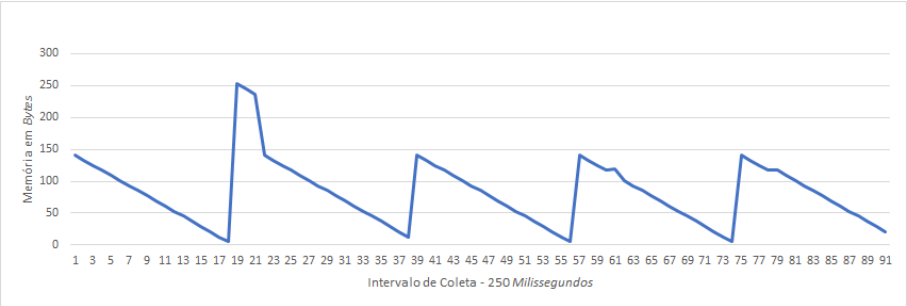
Figura 75 – Memória disponível na transmissão dos dados utilizando sensor de temperatura - Análise 1.



Fonte: Próprio Autor

Em relação aos dados da Figura 75, a análise (Análise 2) da Figura 76 ilustra o consumo de memória utilizando também um sensor de temperatura, mas, desta vez é feito com que o sensor sofra algumas variações para que se fosse observado o consumo de memória. O tempo de coleta de cada informação de memória foi realizada em intervalos de 250 millissegundos. É observado que mesmo consumindo um pouco mais de memória com as variações provocadas, o sistema acabou por gerenciar o consumo de memória e não permitindo estouro de pilha (*overflow*) de memória da aplicação.

Figura 76 – Memória disponível na transmissão dos dados utilizando sensor de temperatura - Análise 2.



Fonte: Próprio Autor

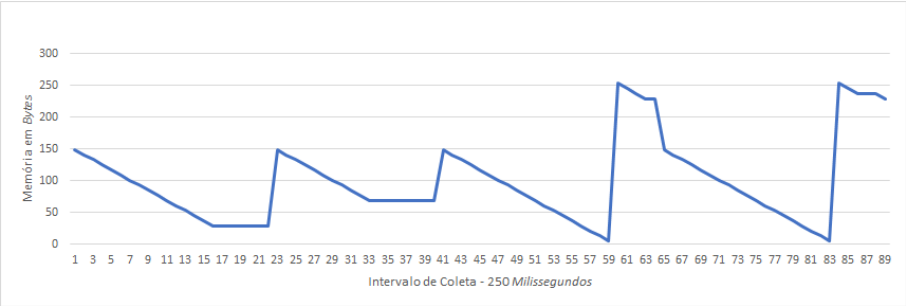
Na análise (Análise 3) da Figura 77 é observado o controle de memória através do sensor de temperatura. As coletas de memória utilizada é em intervalos de 250 millisegundos. Verifica-se que o gerenciamento da informação aconteceu de forma satisfatória e que as variações na temperatura, o consumo de memória permaneceu sobre controle evitando o estouro de pilha da aplicação.

Figura 77 – Memória disponível na transmissão dos dados utilizando sensor de temperatura - Análise 3.

Fonte: Próprio Autor

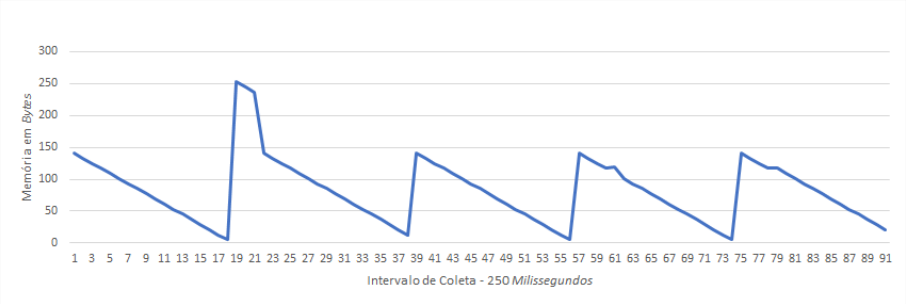
Nas análises (Análise 1 e 2) das Figuras 78 e 79 respectivamente, onde eram gerados valores de temperaturas aleatórias através de uma função na linguagem de programação C/C++, é observado que mesmo realizando o controle e gerenciamento de memória, ocorreu estou de pilha em um tempo muito maior do que sem o uso da técnica conforme ilustrado na Figura 80.

Figura 78 – Memória disponível na transmissão dos dados utilizando sensor de temperatura com dados aleatórios - Análise 1.



Fonte: Próprio Autor

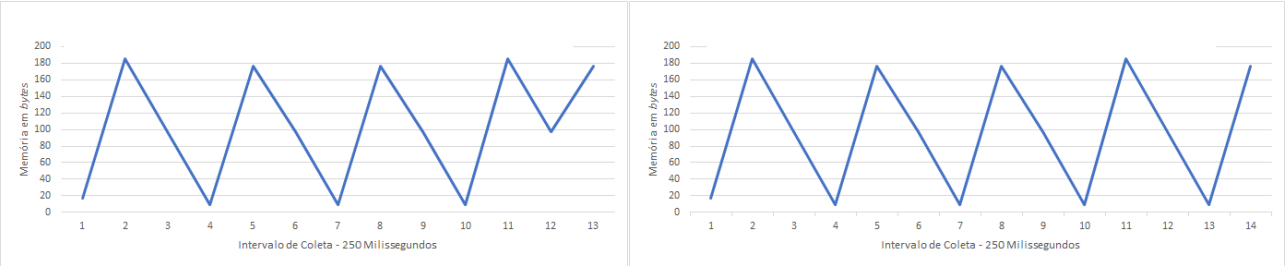
Figura 79 – Memória disponível na transmissão dos dados utilizando sensor de temperatura com dados aleatórios - Análise 2.



Fonte: Próprio Autor

Na Figura 80 pode-se observar que a utilização de memória do sistema sem a utilização da tabela de dispersão ocorre em aproximadamente em 4 segundos. Vale ressaltar que os dados analisados, foram obtidos através de um sensor de temperatura. Como a variação da temperatura ocorre de forma demorada, os valores e variações dos gráficos são praticamente iguais. A Figura 80 (a) e Figura 80 (b) comparada com a Figura 76 evidencia que o sistema composto com a técnica de tabelas de dispersão consome menos memória, já no exemplo sem a utilização da técnica o estouro de pilha (*overflow*) ocorreu em aproximadamente em 4 intervalos de 250 milissegundos cada um.

Figura 80 – Memória disponível sem utilização de tabelas de dispersão.



(a) (b)

Fonte: Próprio Autor

Na Tabela 5 é relatada o consumo de memória de cada análise realizada e comprovando a eficácia da utilização de tabelas *hash* no gerenciamento de memória.

Tabela 5 – Comparativo de Memória Consumida nas Análises

	Temp. com *Hash*	**Temp. Aleatória com *Hash***	**Sem *Hash***
	Análise 1	***Análise 1***	***Análise 1***
Memória - Bytes	*140 - 180*	*150 - 0*	*180 - 0*
250 Milissegundos	****	*59 - 23*	*4*
	Análise 2	***Análise 2***	***Análise 2***
Memória - Bytes	*125 - 140*	*150 - 0*	*180 - 0*
250 Milissegundos	****	*18 - 20*	*4*

Os dados da Tabela 5 evidenciam que a integração do microcontrolador com os protocolos CAN e *ZigBee* obtiveram resultados expressivos no gerenciamento de memória. A tabela relata que os dados do sensor de temperatura LM35 coletados e gerenciados à partir da implementação da tabela de dispersão proposta mostra que a técnica evitou o estouro de pilha (falta de memória) da aplicação. Já o problema utilizando-se da simulação de um sensor que coletava valores capturados de forma aleatória em um intervalo de 250 milissegundos mostraram que estouro de pilha mas em tempos muitos superiores do que a implementação sem a utilização da técnica de tabelas de dispersão. Na coleta aleatória utilizando a tabela de dispersão houve estoure de pilha no intervalo de 59 períodos de 250 milissegundos e também em 23 períodos de 250 milissegundos (Análise 1). Para a Análise 2 existiu um estouro de pilha no intervalo de 18

períodos de 250 milissegundos e outro no vigésimo intervalo de 250 milissegundos. O problema sem a utilização da tabela de dispersão fez com que houvesse estouro de pilha no quarto período de 250 milissegundos. Assim, verifica-se que para problemas onde não há muita variação na leitura de um sensor (temperatura) o sistema se mostra muito eficaz, em problemas onde os valores lidos pelos sensores sofre muita variação (direção do vento) o sistema se mostrou mais eficiente do que em um problema que não utilizou a técnica das tabelas de dispersão. A média para a Análise 1 com a utilização de tabelas de dispersão obteve um desempenho superior de 1025% se comparada a análise que não utilizou a tabela *hash*, já para a Análise 2 o desempenho ficou na margem de 475%. Assim, o ganho médio de desempenho foi em média de 750% com a utilização das tabelas de dispersão para o gerenciamento de memória do microcontrolador.

7.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Nesta seção verifica-se o funcionamento da integração dos protocolos CAN e IEEE 802.15.4 na transmissão e comunicação de dados de uma rede cabeada (CAN) para uma rede sem fio (“ZigBee”). Destaca-se que mesmo com a limitação de memória do microcontrolador *Atmega*328, os algoritmos implementados funcionaram adequadamente para o controle de uma rede de sensores sem fio. O ganho com a utilização das tabelas de dispersão proposta nesta tese chegou a ser 750% superiores no gerenciamento de memória do que as que não se utilizam das tabelas de dispersão.

As análises de tempo para inserir, buscar e remover dados, além da quantidade de memória utilizada efetuadas neste trabalho não foram comparadas com os da literatura, pois, não foram localizados trabalhos na literatura que permitisse tal comparação.

8 CONSIDERAÇÕES FINAIS

8.1 CONCLUSÕES

Neste trabalho apresentou-se o desenvolvimento de um nó de rede heterogêneo (híbrido) em um único microcontrolador que é capaz de integrar o protocolo CAN e o padrão de rede sem fio IEEE 802.15.4. Este nó heterogêneo é capaz de receber dados de uma rede CAN, processar os dados convertendo para um padrão IEEE 802.15.4 e enviar as informações para outros pontos da rede através de comunicação sem fio. Uma das principais características do nó de rede desenvolvido é ser totalmente embarcado e autônomo, realizando o roteamento de cada pacote, baseado em uma de suas normas do padrão IEEE 802.15.4. Além de ser embarcado, o hardware foi desenvolvido baseado nas diferentes taxas de transmissão de dados dos dois protocolos, minimizando os custos em relação a dispositivos e processamento de recursos não utilizados na aplicação. Tais características foram validadas no trabalho, sendo que, para o desenvolvimento do nó de rede, foram utilizados somente os periféricos necessários, minimizando custos do projeto e, otimizando o tempo de operação das tarefas do sistema, tais como, operações de inserção, busca e remoção de dados e diminuição da memória consumida pelo sistema.

A utilização de microcontroladores para a integração de diferentes protocolos de comunicação não é usual, pois, a limitação de recursos em sistemas embarcados torna esta integração lenta e de difícil implementação. Atualmente as tecnologias de programação permitem desenvolver sistemas com todas as capacidades e com a flexibilidade dos processadores embarcados e funções personalizadas.

A implementação de módulo que integra dois protocolos em um dispositivo dedicado facilita o desenvolvimento de outros projetos, como, por exemplo no roteamento de dados em *switches* e tabelas de roteamento, além de aplicações que necessitem de melhor desempenho no acesso a memória e não possuem alto poder de processamento para isto, como é o caso dos sistemas microcontrolados.

A escolha da rede ZigBee para a transmissão dos dados capturados pela rede CAN justifica-se pelo fato de possibilitar a construção de uma rede de baixo custo, confiável difundida em várias áreas e ser padronizada pela IEEE 802.15.4. Uma outra característica que deve ser ressaltada é que tanto a rede CAN como a rede ZigBee podem ser substituídas por redes de mesma função, pois a unificação de outros protocolos em um mesmo sistema embarcado, que utiliza-se

de tabelas de dispersão para sincronizar as informações de padrões que possuem taxas de transmissão, funcionará perfeitamente. Assim, os conceitos de tabelas de dispersão não necessitam de hardware específico para o seu funcionamento.

Uma característica que deve ser ressaltada é que para a sincronização dos dados entre os dois protocolos foi utilizado dois *threads* (múltiplas linhas executando em paralelo) , um responsável pelo inserção dos dados na tabela de dispersão e o outro pelo envio das informações à estação base através de comunicação sem fio. Neste trabalho utilizou-se a linguagem C/C++, que oferece suporte ao acesso aos periféricos e muito utilizada para a programação de microcontroladores. Ambos os protocolos possuem comunicação serial para o envio das informações. As requisições de monitoramento e controle foram realizadas com sucesso, validando o resultado final do projeto.

8.2 CONTRIBUIÇÕES

Uma contribuição importante do trabalho foi a implementação de um método capaz de integrar e gerenciar memória em aplicações embarcadas com alto desempenho através de tabelas de dispersão, fornecendo uma flexibilidade ao sistema, ampliando a área de pesquisa e apresentando uma nova metodologia de desenvolvimento para a integração de protocolos distintos em um único microcontrolador. Esta pode ser considerada a contribuição principal do trabalho, entretanto, há uma série de contribuições específicas tais como:

1. O desenvolvimento de um nó de rede com dois protocolos aumenta a flexibilidade do sistema ampliando as áreas de aplicação;

2. O desenvolvimento de uma arquitetura focada para redes de transdutores inteligentes, utilizando a norma IEEE 802.15.4 e o padrão CAN;

3. O desenvolvimento de uma nova versão do *software* utilizando conceitos de programação estruturada, ponteiros e *Threads*;

4. A implementação dos nós de rede facilita a implementação de novos nós, com protocolos diferentes dos que foram utilizados no projeto;

5. Utilização do conceito de tabelas de dispersão para melhorar o desempenho no gerenciamento e acesso à memória em sistemas que possuem limitação de recursos como memória e processador.

8.3 Sugestões de Trabalhos Futuros

- Implementação da interface que receba como conexão vários sensores na rede CAN;
- Implementação da interface de configuração remota dos parâmetros dos sensores da rede;
- Implementação da rede de transdutores inteligentes utilizando conceitos de tabelas de dispersão;
- Implementação da interface *plug and play* para detecção dos sensores adicionados à rede;
- Implementação de módulo de expansão de memória de massa para ser utilizado pela tabela de dispersão do sistema;
- Desenvolvimento de um software para capturar e armazenar as informações coletadas pela rede de sensores e, armazená-los em um sistema de banco de dados;
- Desenvolvimento de uma rede *mesh* utilizando módulos *XBee Series* 2 como, coordenadores, roteadores e *end-devices.*
- Integração de outros protocolos de redes industriais com outros padrões de comunicação sem fio (*wireless*) em um mesmo protocolo através da utilização de tabelas de dispersão.

REFERÊNCIAS

ATMEL Corporation. [S.l: s.n], 2014. Disponível em: <http://www.atmel.com.br>. Acesso em: 11 jul. 2015.

BARBER, R.; LOHMAN, G.; PANDIS, I.; RAMAN, V.; SIDLE, R.; ATTALURI, G.; CHAINANI, N.; LIGHTSTONE, S.; SHARPE, D. Memory-efficient hash joins. *Proceedings of the VLDB Endowment*, v. 8, n. 4, p. 353–364, 2014. Disponível em: <http://dl.acm.org/citation.cfm?id=2735496.2735499>. Acesso em: 10 fev. 2015.

BORGES, F. *Transmissão de dados:* documento técnico n. 3. [s.l.: s.n.]. 2008. Disponível em: <http://www.schneiderelectric.pt/documents/product-services/training/transmissao-dados.pdf>. Acesso em: 11 maio 2015.

BOSCH. *CAN specification version 2.0*. [S.l.]: Bosch, 1991. Disponível em: <https://books.google.com.br/books?id=hvg4HAAACAAJ>. Acesso em: 5 jun 2012.

BRUDNA, C. *Desenvolvimento de sistemas de automação industrial baseados em objetos distribuídos e no barramento CAN*. Porto Alegre: [s. n.], 2000.

BYRES, E. J. Designing secure networks for process control. In: *INDUSTRY TECHNICAL CONFERENCE RECORD, 1999. Annual...* : [S.l.]: Pulp and Paper, 1999. p. 63–67.

CELES, W.; CERQUEIRA, R.; RANGEL, J. *Introdução a estruturas de dados:* com técnicas de programação em c. Rio de Janeiro: Campus, 2004. Disponível em: <https://books.google.com.br/books?id=_TYWOgAACAAJ>. Acesso em: 12 mar 2015.

CHEN, D.; MOK, A. K. *Developing new generation of process control systems.* [S.l.: s. n.], 2001.

COMMITTEE, L. S. et al. *Part 15.4:* wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). Washington: IEEE Computer Society, 2003.

DJIEV, S. *Industrial networks for communication and control*. [S.l.: s. n.], 2003. Disponível em: <http://anp.tu-sofia.bg/djiev/PDF%20files/Industrial%20Networks.pdf>. Acesso em: 11 mar 2013.

ELETROGATE. *Arduino básico V1.0*. [S.l.: s. n.], 2015. Disponível em: <http://apostilas.eletrogate.com/Apostila_Arduino_Basico-V1.0-Eletrogate.pdf>. Acesso em: 11 jul 2015.

FILHO, C. S.; FINKEL, V. Sistemas de automação e adequação funcional dos profissionais de automação e TI industrial. *Revista InTech*, São Paulo, n. 51, p. 24–28, 2003.

FORTE, M. Protocolos de comunicação: analisar e só depois escolher. *Revista Controle e Instrumentação*, Osasco, n. 94, p. 54–59, 2004.

GAO, H.; GROOTE, J. F.; HESSELINK, W. H. Lock-free dynamic hash tables with open addressing. *Distributed Computing*, Heidelberg, v. 18, n. 1, p. 21–42, 2005.

GUEDES, L. A. *Classificação das redes para automação industrial*. [S.l.: s.n.], 2005. Disponível em: <http://www.dca.ufrn.br/˜affonso/DCA0447/aulas/rai_cap3_part1.pdf>. Acesso em: 14 out 2014.

GUTIERREZ, J. A.; NAEVE, M.; CALLAWAY, E.; BOURGEOIS, M.; MITTER, V.; HEILE, B. IEEE 802.15. 4: a developing standard for low-power low-cost wireless personal area networks. *IEEE Network*, Piscataway, v. 15, n. 5, p. 12–19, 2001.

GUTIERREZ, R. M. V.; PAN, S. S. K. *Complexo eletrônico:* automação do controle industrial. [S.l.: s.n.], 2008. Disponível em: <https://web.bndes.gov.br/bib/jspui/handle/1408/9536>. Acesso em: 4 ago 2014.

HENNESSY, J. L.; PATTERSON, D. A. *Computer architecture:* a quantitative approach. [S.l.]: Elsevier, 2011.

HUANG, H.-W. *MC68HC12 An Introduction:* software and hardware interfacing. [S.l.]: Cengage Learning, 2003.

HUBERT, M. K. O protocolo CAN como solução para aplicações distribuídas, baseadas em objetos, entre pcs e microcontroladores. Pelotas: Universidade Federal de Pelotas, 2001.

International Telecommunications Union, I. *Birth of broadband.* [S.l.: s.n.], 2003. Disponível em: <http://www.itu.int/osg/spu/publications/sales/birthofbroadband/>. Acesso em: 11 jun 2015.

KONGEZOS, V. K.; ALLEN, C. R. Wireless communication between agvs (autonomous guided vehicles) and the industrial network can (controller area network). In: *INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION- ICRA, 2002. Proceedings...* : [S.l.: s.n.], 2002. v. 1, p. 434–437.

KUBAN, P. A. *An architecture for the extension of fixed controller area networks to ieee 802.15. 4 wireless personal area networks*. Louisville: University of Louisville, 2006.

KUMAR, S.; CROWLEY, P. Segmented hash: an efficient hash table implementation for high performance networking subsystems. In: . : SYMPOSIUM ON ARCHITECTURES FOR NETWORKING AND COMMUNICATIONS SYSTEMS - ANCS, 2005. *Proceedings...* [S.l.: s.l.], 2005. p. 91–103.

KUROSE, J. F. *Computer networking:* a top-down approach featuring the internet, 3/E. Boston: Pearson Education, 2005.

MONTEZ, C. *Redes de comunicação para automação industrial*. Florianópolis: Universidade Federal de Santa Catarina, 2005. Disponível em: <http://carlosmontez.paginas.ufsc.br>. Acesso em: 12 set 2009.

MORAES, C. de; CASTRUCCI, P. de L. *Engenharia de automação industrial*. Rio de Janeiro: LTC, 2007. Disponível em: <https://books.google.com.br/books?id=irN1PgAACAAJ>. Acesso em: 10 abr 2014.

NOGUEIRA, T. A. *Redes de comunicação para sistemas de automação industrial*. Ouro Preto: Universidade Federal de Ouro Preto, 2009. Disponível em: <http://www.em.ufop.br/cecau/monografias/2009/THIAGO%20AUGUSTO.pdf>. Acesso em: 10 maio 2014.

POPP, M. *The rapid way to PROFIBUS-DP*. PROFIBUS-nutzerorganisation, [S.l.: s.n.], 1997.

PREISS, B. *Estruturas de dados e algoritmos:* padrões de projetos orientados a objetos com java. Rio de Janeiro: Campus, 2001. Disponível em: <https://books.google.com.br/books?id=stM7AAAACAAJ>. Acesso em: 12 mar 2013.

REAVES, B.; MORRIS, T. Analysis and mitigation of vulnerabilities in short-range wireless communications for industrial control systems. *International Journal of Critical Infrastructure Protection*, Amsterdam, v. 5, n. 3, p. 154–174, 2012.

RENDA, M. E.; RESTA, G.; SANTI, P. Load balancing hashing in geographic hash tables. *IEEE Transactions on Parallel and Distributed Systems*, Piscataway, v. 23, n. 8, p. 1508–1519, Aug 2012.

SALAMI, B.; ARCAS-ABELLA, O.; SONMEZ, N. *HATCH: Hash table caching in hardware for efficient relational join on FPGA*. : In: ANNUAL INTERNATIONAL SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES - FCCM, 23., 2015. *Proceedings...* [S.l.]: Institute of Electrical and Electronics Engineers, 2015. 163 p.

STANDARD, I. *11898:* road vehicles-interchange of digital information - controller area network (can) for high-speed communication. [S. l.]: International Standards Organization, Switzerland, 1993.

STEPANOV, M.; ROSNER, G. *Dynamic customization and adaptive content delivery for wireless clients.* [S.l.: s.n.], 2002. Disponível em: <http://www.cs.huji.ac.il/course/2002/sdbi/2003/wireless.ppt>. Acesso em: 17 nov 2015.

SZWARCFITER, J.; MARKENZON, L. *Estruturas de dados e seus algoritmos*. [S.l.]: Livros Tecnicos e Cientificos, 1994.

TAKENAKA, Y.; SENO, S.; MATSUDA, H. Perfect hamming code with a hash table for faster genome mapping. *BMC genomics*, London, v. 12, n. 3, p. S8, 2011. Suplemento. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/22369457>. Acesso em: 5 fev 2014.

TANENBAUM, A. *Computer networks*. [S. l.]: Prentice Hall PTR, 2003. (Computer Networks, p. 3). Disponível em: <https://books.google.com.br/books?id=0hkZAQAAIAAJ>. Acesso em: 19 ago 2013.

TANENBAUM, A. *Redes de computadores*. Rio de Janeiro: CAMPUS, 2003. Disponível em: <https://books.google.com.br/books?id=0tjB8FbV590C>. Acesso em: 28 jul 2015.

TONG, D.; ZHOU, S.; PRASANNA, V. K. High-throughput online hash table on fpga. In: . : INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOPS - IPDPSW, 29., 2015. *Proceedings...* [S.l.]: Institute of Electrical and Electronics Engineers, 2015. p. 105–112.

WEG. Automação de processos industriais: módulo 3. [S.l.: s.n.], 2002. Disponível em: <ftp://ftp.mecanica.ufu.br/LIVRE/SCHP/arquivos/Automa%E7%E3o%20de%20Processos%20Industriais%20-%20WEG.pdf>. Acesso em: 14 out 2016.

WILD, R.; PEREIRA, C. A tool for validating timing requirements of industrial applications based on the foundation feldbus protocol. *IFAC Proceedings*, Kidlington, v. 32, n. 1, p. 33–38, 1999.

XCTU. Digi. [S.l.: s.l.], 2014. Disponível em: <https://www.digi.com/>. Acesso em: 01 maio 2014.

XIONG, B.; LI, F.; JIANG, L.; CHEN, X. An efficient oriented-hash table for connection record management in high-speed networks. *Huazhong Keji Daxue Xuebao (Ziran Kexue Ban)/Journal of Huazhong University of Science and Technology (Natural Science Edition)*, Wuhan, v. 39, n. 2, p. 19–22+31, 2011.

APÊNDICE A – ALGORITMOS ELABORADOS NO TRABALHO

No Apêndice A estão definidos os códigos desenvolvidos na tese. No Apêndice A.1 é definido o algoritmo que captura a informação de um sensor de temperatura e as envia pelo barramento CAN. No Apêndice A.2 está definido o algoritmo do módulo que integra os protocolos CAN e IEEE 802.15.4 em um único microcontrolador. No Apêndice A.3 é descrito o código da Estação Base da aplicação que coleta as informações da rede sem fio.

APÊNDICE A.1 - Módulo CAN

```
// demo: CAN-BUS Shield, send data
#include <mcp_can.h>
#include <SPI.h>

// the cs pin of the version after v1.1 is default to D9
// v0.9b and v1.0 is default D10
const int SPI_CS_PIN = 9;
const int ledHIGH    = 1;
const int ledLOW     = 0;

// Define o pino que lera a saída do LM35
const int LM35 = A0;

// Variável que armazenará a temperatura medida
long temperatura;

unsigned char stmp[8] = {0, 1, 2, 3, 4, 5, 6, 7};

// Set CS pin
MCP_CAN CAN(SPI_CS_PIN);

void setup()
{
        Serial.begin(9600);
        randomSeed(analogRead(0));
        START_INIT:

        // init can bus : baudrate = 1000k
        if(CAN_OK == CAN.begin(CAN_1000KBPS))
        {
                Serial.println("CAN BUS Shield Iniciado!");
        }
        else
        {
                Serial.println("CAN BUS Shield Falhou ao Iniciar");
                Serial.println("Iniciar CAN BUS Shield Novamente");
```

```
37                  delay(100);
38                  goto START_INIT;
39          }
40  }
41
42  //Sensor de temperatura usando o LM35
43
44  void loop()
45  {
46
47          //Calcula a temperatura lida pelo sensor LM35
48          temperatura = int((float(analogRead(LM35))*5/(1023))/0.01);
49
50          //temperatura = random(47);
51          stmp[0] = temperatura;
52
53          //Envia informação no Barramento CAN
54          CAN.sendMsgBuf(0x70,0, 8, stmp);
55
56          //Envia dados a cada 250 milissegundos
57          delay(250);
58  }
59
60  /*********************************************************
61  END FILE
62  *********************************************************/
```

APÊNDICE A.2 - Módulo Integrador CAN com IEEE 802.15.4

```
1
2  // Módulo Integrador entre os Protocolos CAN e IEEE 802.15.4
3
4
5  #include <SPI.h>
6  #include "mcp_can.h"
7  #include <math.h>
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include "Thread.h"
11 #include "ThreadController.h"
12 #define MALLOC(x) ((x *) malloc (sizeof(x)))
13
14 ThreadController cpu;
15
16 Thread inserir;
17 Thread remover;
18
19 //declaração da estrutura
20 struct no{
21         byte valor;
22         byte quantidade;
23         struct no *proximo;
24 };
25
26 typedef struct no no_t;
27
28 byte valor;
29
30 // the cs pin of the version after v1.1 is default to D9
31 // v0.9b and v1.0 is default D10
32 const byte SPI_CS_PIN = 9;
33 const byte LED = 8;
34 boolean ledON = 1;
35
36 // Set CS pin
```

```
37 MCP_CAN CAN(SPI_CS_PIN);
38
39 //Armazena os valores recebidos da serial
40 byte valores = 0;
41 //Armazena o estado do led
42 String estado;
43
44 //Estrutura da Tabela de Dispersão
45 no_t dados[47];
46
47 no_t *fim;
48
49 //Cria Inicio da Lista
50 no_t *inicio = NULL;
51
52 //declaração das funções
53 no_t *cria();
54 void *insini(no_t *lista, byte valor);
55 void *insfim(no_t *lista, byte valor);
56 void *retira();
57
58 //Protótipo da Função que informa a memória disponível
59 byte freeRam ();
60
61 //Função que cria o nó
62 no_t *cria(){
63         inicio = NULL;
64 };
65 //************************************************************
66
67 //Função que remove o elemento enviado
68 void *retira(byte valor)
69 {
70
71         no *tmp;
72         int x;
73         tmp = inicio;
74         x = tmp->valor;
```

```
75          if(dados[valor%47].quantidade == 1)
76          {
77                  inicio = tmp->proximo;
78                  dados[valor%47].quantidade = dados[valor%47].
                        quantidade-1;
79
80                  free (tmp);
81          }
82          else
83          {
84                  tmp->quantidade = tmp->quantidade-1;
85                  dados[valor%47].quantidade = dados[valor%47].
                        quantidade-1;
86          }
87
88          Serial.print("Valor Enviado:");
89          Serial.print(x);
90  };
91
92
93  //Função para inserir Elemento na estrutura
94  void *insini(no_t *lista, byte valor)
95  {
96          no_t *novo;
97
98          novo = MALLOC(no_t);
99
100         novo->valor = valor;
101         novo->quantidade = 1;
102         novo->proximo = NULL;
103
104         if(inicio == NULL)
105         {
106                 inicio = novo;
107                 dados[valor%47].quantidade = 1;
108                 dados[valor%47].valor = valor;
109         }
110         else
```

```
111         {
112                 if (inicio != NULL)
113                 {
114                         fim = inicio;
115                         while (fim->proximo != NULL)
116                         {
117                                 fim = fim->proximo;
118                         }
119
120                         if(fim->valor == valor)
121                         {
122                                 fim->quantidade = fim->quantidade
                                       +1;
123                         }
124                         else
125                         {
126                                 fim->proximo = novo;
127                         }
128                 }
129
130                 dados[valor%47].quantidade = dados[valor%47].
                       quantidade+1;
131                 dados[valor%47].valor = valor;
132                 //dados[valor%47].proximo = novo;
133         }
134 };
135
136 void parte1()
137 {
138         // check if data coming
139         if(CAN_MSGAVAIL == CAN.checkReceive())
140         {
141                 // read data,  len: data length, buf: data buf
142                 CAN.readMsgBuf(&len, buf);
143
144                 unsigned char canId = CAN.getCanId();
145
146                 Serial.println("-----------------------------");
```

```
147                 Serial.print("CAN ID: ");
148                 Serial.print(canId);
149                 Serial.print("\n");
150 
151                 valor = buf[0];
152                 x = valor % 47;
153                 Serial.print("Temp.: ");
154                 Serial.println(valor);
155                 insereini(FILA, valor);
156         }
157 }
158 
159 void parte2()
160 {
161         if(inicio != NULL)
162         {
163                 retira(valor);
164         }
165 
166         Serial.print("Memoria: ");
167         Serial.print(freeRam());
168         Serial.println("\n");
169 
170 }
171 
172 //*********************************************************
173 
174 void setup()
175 {
176         //Configuração dos Módulos
177         xbee.setSerial(Serial);
178         inserir.setInterval(250);
179         inserir.onRun(parte1);
180         remover.setInterval(1000);
181         remover.onRun(parte2);
182         cpu.add(&inserir);
183         cpu.add(&remover);
184 
```

```
185         FILA = (node *) malloc(sizeof(node));
186         if(!FILA){
187                 //    printf("Sem memoria disponivel!\n");
188                 exit(1);
189         }else{
190                 inicia(FILA);
191         }
192
193         randomSeed(analogRead(0));
194         Serial.begin(9600);
195         pinMode(LED, OUTPUT);
196         //Define o pino 13 - LED embutido no Arduino - como saida
197         pinMode(13, OUTPUT);
198         Serial.println("Aguardando Dados!!! A serem enviados pelo
               ZigBee...");
199
200
201         START_INIT:
202
203         // init can bus : baudrate = 1000k
204         if(CAN_OK == CAN.begin(CAN_1000KBPS))
205         {
206                 Serial.prbyteln("CAN BUS Shield Iniciado!");
207         }
208         else
209         {
210                 Serial.prbyteln("CAN BUS Shield Falhou");
211                 delay(100);
212                 goto START_INIT;
213         }
214 }
215
216 //*********************************************************************
217 byte freeRam () {
218         extern byte __heap_start, *__brkval;
219         byte v;
220         return (int) &v - (__brkval == 0 ? (int) &__heap_start : (
               int) __brkval);
```

```
221 }
222
223 //*************************************************************
224
225
226 void loop()
227 {
228         cpu.run();
229         Serial.flush();
230         Serial.println(freeRam());
231         delay(250);
232 }
233
234 /*****************************************************************
235 END FILE
236 *****************************************************************
    */
```

APÊNDICE A.3 - Módulo da Estação Base - ZigBee

```
#include <XBee.h>
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();


void setup() {
        Serial.begin(9600);
        xbee.begin(Serial);
}

void loop() {
        byte sample;
        xbee.readPacket();

        if (xbee.getResponse().isAvailable()) {
                Serial.print("ID. Modulo XBEE: ");
                Serial.println(xbee.getResponse().getApiId());

                if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE
                   ) {
                        xbee.getResponse().getZBRxResponse(rx);
                        for (int i = 0; i < rx.getDataLength(); i
                           ++) {
                                sample = rx.getData(i);
                        }

                        Serial.print("Temperatura: ");
                        Serial.print(sample);
                        Serial.println("C");
                }

        }else if (xbee.getResponse().isError()) {
                Serial.println("Erro na Leitura do Pacote.  Código
                   do Erro: ");
                Serial.println(xbee.getResponse().getErrorCode());
```

```
34          }
35          delay(100);
36  }
```

APÊNDICE B – ARTIGOS PUBLICADOS EM CONGRESSOS E REVISTAS

No Apêndice B estão listados todos os artigos publicados em congressos e revistas relacionados a tese desenvolvida.

- Artigo submetido para a revista *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING* em 2017. O título do artigo é: ***Network sensor with hybrid communication (CAN and ZigBee) for industrial applications.***; Autores: Marcos Antonio Estremote; Tiago da Silva Almeida; Nobuo Oki. DOI: 10.1002/wcm. Qualis A2.

- Artigo publicado no congresso Dincon 2015 (*Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*). O título do artigo é: **Transmissão de Imagens Bitmap e Vídeo utilizando o Padrão Zig Bee IEEE 802.15.4**; Autores: Virgílio Fries Müller; Marcos Antonio Estremote; Nobuo Oki.

- Artigo publicado no congresso SBAI 2013. O título do artigo é: ***USING HASH TABLES FOR SYNCHRONIZATION IN COMMUNICATION BETWEEN THE CAN PROTOCOL AND IEEE 802.15.4***. Autores: Marcos Antonio Estremote; Nobuo Oki.

- Artigo publicado no congresso Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE. Título do Artigo: ***Development of Heterogeneous Node Network Using Microcontroller with Protocols CAN and Zigbee***. Autores: Marcos Antonio Estremote; Nobuo Oki. DOI: 10.1109/CERMA.2011.69. Print ISBN: 978-1-4577-1879-3. Electronic ISBN: 978-0-7695-4563-9. Print on Demand(PoD) ISBN: 978-1-4577-1879-3.