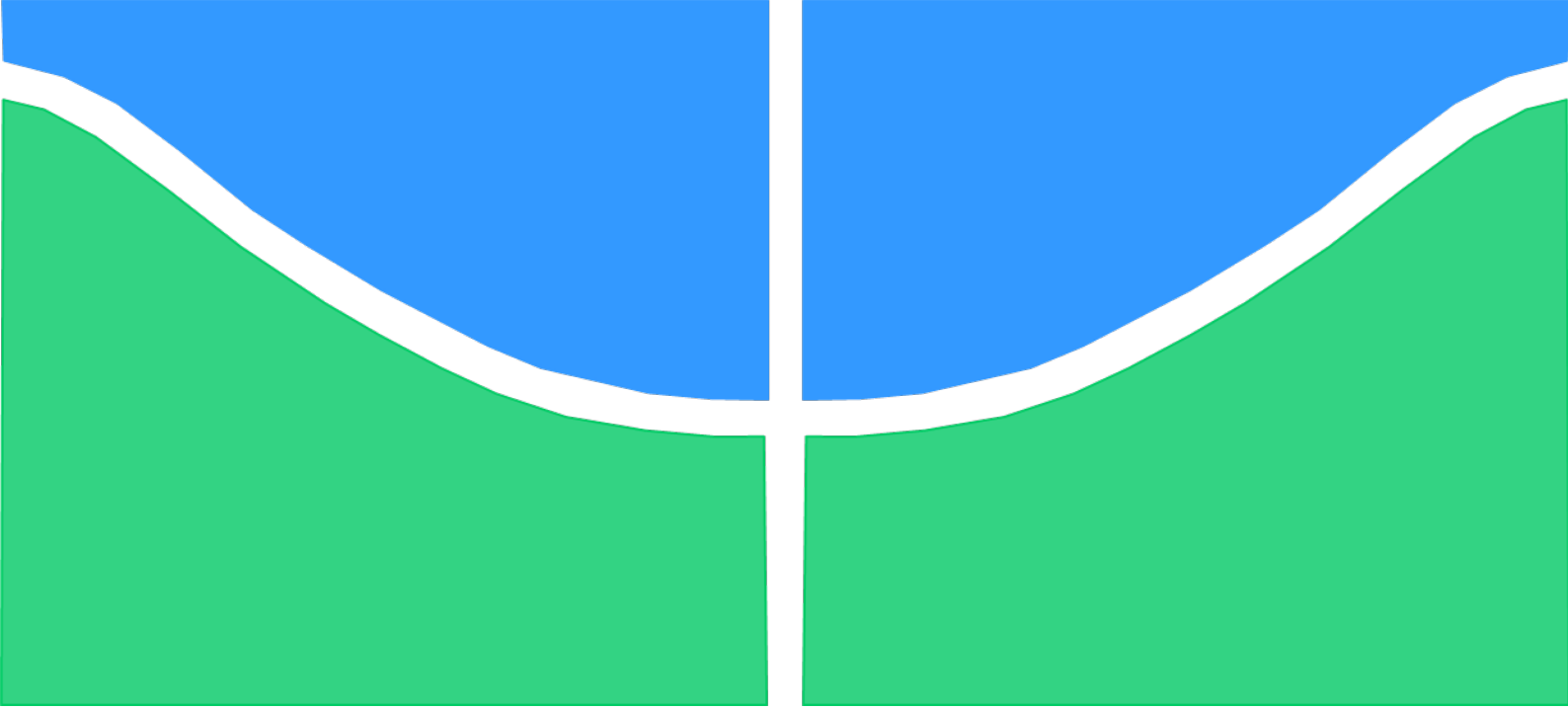

TRABALHO DE GRADUAÇÃO

TRANSFORMADA DE SCATTERING APLICADA AO CONTROLE DE SISTEMAS EM REDE COM ATRASOS VARIADOS

Por,

Vinícius Fernandes de Souza

Franklin de Souza Conceição

Brasília, julho de 2015

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

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

TRABALHO DE GRADUAÇÃO

TRANSFORMADA DE SCATTERING APLICADA AO CONTROLE DE SISTEMAS EM REDE COM ATRASOS VARIADOS

Por,
Vinícius Fernandes de Souza
Franklin de Souza Conceição

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

Banca Examinadora

Prof. Adolfo Bauchspiess, ENE/UnB _____
Orientador

Prof. Flavia Maria Aranha Oliveira, ENE/UnB _____
Examinador Interno

Prof. Lélio Ribeiro Soares Júnior, ENE/UnB _____
Examinador Interno

Brasília, julho de 2015

FICHA CATALOGRÁFICA

SOUZA, V. F., CONCEIÇÃO, F. S.
TRANSFORMADA DE SCATTERING APLICADA AO CONTROLE DE SISTEMAS EM REDE COM ATRASOS VARIADOS

[Distrito Federal] 2015.

viii, 93p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Automação
2.Scattering
3.Zigbee

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

V. F. SOUZA, F. S. CONCEIÇÃO, (2015). TRANSFORMADA DE SCATTERING APLICADA AO CONTROLE DE SISTEMAS EM REDE COM ATRASOS VARIADOS. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 01, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 93p.

CESSÃO DE DIREITOS

AUTOR: V. F. SOUZA, F. S. CONCEIÇÃO

TÍTULO DO TRABALHO DE GRADUAÇÃO: TRANSFORMADA DE SCATTERING APLICADA AO CONTROLE DE SISTEMAS EM REDE COM ATRASOS VARIADOS.

GRAU: Engenheiro de Controle e Automação ANO: 2015

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

Dedicatórias

Dedico a todos os que me apoiaram e que de alguma forma me deram força pra continuar

Franklin de Souza Conceição

Dedico a universidade e atodos os docentes que me apoiaram ao longo do curso.

Vinícius Fernandes de Souza

Agradecimentos

Agradeço a Deus e a todos, sem exceção, que me apoiaram e me mantiveram motivado por todo o caminho. Desde, os sábios Vinícius Galvão e o Professor Adolfo, a minha família e amigos.

Vinícius Fernandes de Souza

Agradeço a Deus e a todos que me apoiaram ao longo do curso de graduação, em especial à minha família que sempre esteve ao meu lado, agradeço ao engenheiro Vinícius Galvão por sempre estar à disposição quando foi preciso, agradeço ao professor Adolfo B. que sempre procurou orientar e ajudar no que fosse necessário, por último agradeço à Universidade de Brasília por ter me proporcionado a estrutura necessária e os professores para que eu pudesse alcançar a graduação em engenharia.

Franklin de Souza Conceição

RESUMO

Resumo do trabalho de graduação

Este trabalho de graduação visa utilizar placas ZigBit para implementar uma rede de automação predial não cabeada que apresente robustez e estabilidade, o controle da automação predial é na área de refrigeração, onde por limitações de equipamentos, foi simulado neste trabalho por um sistema de aquecimento. Como controle da planta do sistema foi utilizado um sistema de realimentação de estados, onde usamos três controladores para príncipio de comparação. O objetivo do trabalho consiste em verificar a eficiência do uso da transforamda de Scattering frente aos atrasos da atuação gerados por limitações físicas e pertubações.

Para a implementação deste trabalho foram confeccionadas placas com o microcontrolador ATmega8, com rede de comunicação desenvolvida para esta aplicação baseada em um código exemplo da Atmel. Para aproximar o sistema ao mercado de automação predial, foi implementado um protrocolo de comunicação que utiliza o protocolo BacNet para realizar a comunicação entre as placas.

Para validação do modelo do controlador estudado usou-se a comparação com outros dois modelos: o controlador de Atraso e o Smith Predictor. Para obtenção dos resultados foram aplicadas à planta pertubações afim de variar o atraso da condução térmica afim de verificar o comportamento de cada um destes controladores no sistema.

Palavras Chave: Scattering, Smith Predictor, Automação, Atraso, ZigBee, ZigBit, Atmel, BACnet, NCS.

ABSTRACT

This project aims to use ZigBit boards to implement a non-wired building automation network to present robustness and stability, control of building automation is the cooling area, where by equipment limitations, was simulated in this paper as heating system. As system plant control we used a state feedback system where we use three drivers for comparison principle. The objective is to verify the efficiency of the use of transforamda of Scattering front delays the performance generated by physical limitations and disturbances.

To implement this job boards were made with ATmega8 microcontroller, with communications network developed for this application based on a sample code from Atmel. To bring the system

to the building automation market, it has been implemented a protrocolo communication using BACnet protocol for performing communication between the plates.

To validate the model of the studied controller used to compare with the other two models: Delay controller and Smith Predictor. To obtain the results were applied to plant disturbances in order to vary the delay of thermal conduction in order to check the behavior of each of these controllers in the system.

Keywords: *Scattering, Smith Predictor, Automation, Delay, ZigBee, ZigBit, Atmel, BACnet, NCS.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

k	Constante de Condutividade Térmica	[W/m*K]
A	Área	[m^2]
L	Comprimento	[m]
T	Temperatura	[^oK]
R	Resistência	[Ω]
C	Capacitância	[F]

Grupos Adimensionais

G	Ganho

Subscritos

ref	referência
sis	sistema

Siglas

ADC	Conversor analógico digital
API	*Application programming interface*
APS	Subcamada de aplicação e suporte (*Application Support Sublayer*)
Atmega1281R	Micro controlador usado na placa ZigBit utilizado neste projeto
BACnet	*Building Automation and Control NETworks*
BSP	Biblioteca que implementa as funções de botões e sensors da placa
Fmincon	Função do Matlab para minimização de parâmetros
HAL	Funções que implementam as funcionalidades do hardware do módulo
HVAC	*Heating Ventilator Air Condicioner*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
ITAE	*Integral of Time and Absolute Error*
NCS	*Networked Controll System*
PDS	
PID	Proporcional Integral Derivativo
Td	Atraso real
Tdn	Atraso Nominal
UART	*Universal Asynchronous Receiver/Transmitter*
USART	*Universal Synchronous/Asynchronous Receiver/Transmitter*
VS	Versus
WNCS	Wireless Network System Control
WSNDemo	Programa para download do arquivo de compilação nas placas MeshBean
ZigBee	Protocolo de rede de baixo consume
ZigBit	Módulos *wireless* seriais de protocolo 802.15.4/ZigBee
ZCL	*Zigbee Cluster Library*

Capítulo 1

Introdução

Esse capítulo faz uma introdução dos sistemas de automação, suas especificidades de rede e controle e a análise de soluções em controle dependente e independente de atrasos.

1.1 Contextualização

O desenvolvimento constante das novas tecnologias de controle vem trazendo inovações em vários níveis de aproveitamento das quais a automação predial/residencial vem recebendo uma maior atenção nos últimos anos [17].

Com as diversas mudanças culturais observadas atualmente, a busca pelo conforto e pela comodidade de forma sustentável é uma das mais marcantes na sociedade. Inserida nesse contexto tem-se que a automação vem ganhando seu espaço no mercado e no dia-a-dia da sociedade principalmente com o crescimento e difusão da automação residencial e predial.

A automação predial envolve diversas bases de conhecimento e abrangência, tal que não se limita a um sistema HVAC (Heating, Ventilating and Air Conditioning System). Dentre as áreas de atuação as mais comumente observadas são o Sistema de Controle de Acesso, Sistema de Iluminação, Sistema de Circuito Fechado de TV, Sistema de Detecção e Alarme de Incêndio e Integração de Sistemas,[1].

O projeto de automação predial não se limita a atender um certo problema caracterizado pelo cliente. A ideia é que o sistema se torne aprofundado e que busque não somente automatizar certos processos, mas também obter um custo correlacionado menor, tanto custo financeiro, quanto de recursos humanos e energia, objetivando uma maior eficiência em todos os sentidos.

Para cada um dos sistemas da automação predial, pode-se apresentar soluções que buscam a eficiência. Para que a solução de mercado seja verdadeiramente eficiente, é necessário um estudo de caso que garanta a especificidade e os pré-requisitos do projeto.

A popularidade da automação tem ganhado abrangência nos dias de hoje, tanto por motivos de seguranças quanto por garantias de eficiência energética agregando o valor da ideia do desenvolvimento sustentável[18].

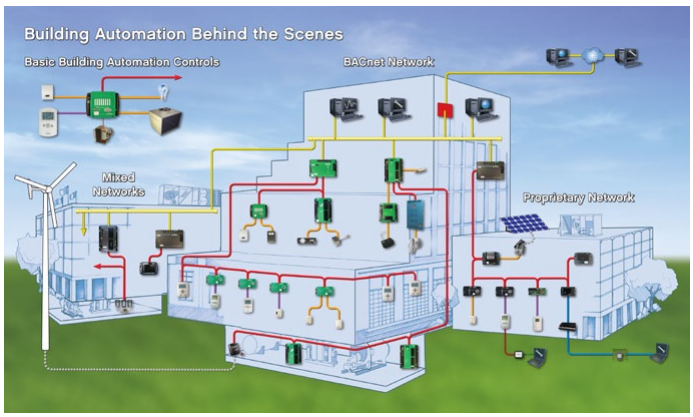

Figura 1.1: Exemplo de projeto de automação predial[1].

O real desafio da automação predial é o controle de demanda, revezamento de equipamentos, programação cronológica, relatórios específicos, dados confiáveis e atualizados, análise de gastos de energia e controle desses gastos[19]. Para implementar uma rede de automação em edifícios utiliza-se em sua maioria o cabeamento estruturado. Porém o cabeamento demanda infraestrutura própria, constantes manutenções, pois cabear um edifício após sua construção tem um custo demasiado elevado (Caso das reformas ou Retrofit). Devido a estas razões o uso de uma rede sem fio para implementar um sistema de automação predial pode baratear a implementação da rede, diminuir o custo de manutenção e diminuir consideravelmente o tempo e a mão de obra empregados para o levantamento da rede, aumentar a flexibilidade e a capacidade de se reconfigurar [20]. Observa-se então que sistemas tradicionais de controle estão sendo substituídos por sistemas no qual a planta e o controlador são conectados por uma rede chamada de Sistema de Controle em Rede (NCS). Entretanto, os atrasos de comunicação, perda de pacotes de dados e limite de recursos, são alguns dos desafios a serem considerados. Dentre as tecnologias que possuem a característica wireless, o protocolo de rede ZigBee destaca-se por suas funcionalidades na área de automação devido ao seu baixo consumo energético e boa taxa de transferência de dados se comparado aos modelos mais famosos como o Bluetooth.

Em meados dos anos 90, início dos anos 2000, os sistemas de automação começaram a ter um grande investimento em decorrência da expansão de sua aplicabilidade em projetos. Com o crescimento veio também a necessidade do uso de ferramentas e equipamentos para implementação de sistemas de automação. Observando essa necessidade da área, um grupo de empresas se uniram e formaram a ZigBee Alliance, que juntamente com o apoio do IEEE desenvolveu um protocolo voltado para aplicações de automação[13].

O ZigBee foi primeiramente criado com o objetivo de atender uma demanda crescente de pesquisadores e empresários que buscavam na automação uma solução clara para seus projetos. Assim, o ZigBee se inseriu no mercado como um protocolo competitivo, open source e multiplataforma[13].

O objetivo da ZigBee Alliance era desenvolver um protocolo de baixo custo de implementação de projeto, baixo custo de uilização (custo de energia muito baixo), simples, expansível, com alto grau de confiabilidade e que permitisse o acesso a todos os interessados no protocolo[13].

O protocolo ZigBee proporcionou uma expansão nos estudos na área de automação utilizando rede wireless. Além disso, devido ao baixíssimo custo de aquisição de equipamentos com padrão ZigBee, essa plataforma se tornou uma das principais escolhas dentre as várias opções de desenvolvimento wireless.

1.2 Definição do problema

Tendo em vista as soluções de automação no mercado observa-se uma certa dependência do sistema a topologias específicas. Por exemplo, quando trata-se de um projeto de automação voltado para a área de ar condicionados tem-se em seu sistema mais simples um controle de temperatura baseado numa temperatura de retorno. Nesse caso, deve-se caracterizar esse sistema como um sistema de controle com um sensor de temperatura que realimenta seu controlador[21].

Ao adicionar elementos de uma solução real, deve-se considerar certas incertezas no controle e no sensoriamento. Quando trata-se de um ambiente simples e fechado o controle de temperatura ótimo será próximo do esperado idealmente, pois o sistema se comporta de maneira que não há variação no atraso ou interferência no mesmo. Entretanto, ao observar uma situação real onde a fonte de refrigeração está em uma determinada sala, e seu sensor deve ser colocado tal que o controle sobre uma sala anexa a esse corredor seja ótimo, tem-se um erro associado que reflete exatamente as incertezas do ambiente como, por exemplo: pessoas caminhando com frequência no corredor, porta da sala anexa fechada ou entre-aberta, captação do sensor em atraso, atrasos variados da rede[22].

Nesse sentido, tem-se a necessidade de garantir uma certa qualidade ou performance do controle em arquiteturas NCS e, para tanto, é necessário algumas abordagens relacionadas ao sistema. Pode-se, por exemplo, utilizar uma rede implementada sob medida para o projeto, tal que ela garanta no mínimo uma faixa de operação do atraso no sistema. Entretanto, ao utilizar redes mais comuns como Ethernet, não se pode gartantir um atraso ou uma faixa de variação desse atraso, e com isso, o sistema se apresenta vulnerável a deterioração e até mesmo instabilidade.

A *Transformada de Scattering* se apresenta como uma possível solução para que se possa utilizar uma arquitetura NCS sem que haja uma preocupação com a deterioração do sinal de controle ocasionada pelo atraso variado[7].

1.3 Objetivos do projeto

O projeto tem por objetivo o desenvolvimento, estudo e pesquisa do tratamento de controle de um NCS em protocolo ZigBee utilizando a *Transformada de Scattering* como metodologia de controle, para lidar com atrasos variáveis e visando uma utilização real no mercado de automação

- O sistema em rede ZigBee será fundamentado numa rede minimamente consistente, ou seja, apesar de ocorrer perda de pacotes, e/ou atrasos a rede não pode ser desfeita ou reduzir o desempenho de controle.

- A metodologia será estudada em comparação com outros dois métodos de controle de rede: Controlador Atraso otimizado e e *Smith Predictor*.
- O sistema deve buscar uma adaptação ao mercado atual, assim, propusemos uma comunicação baseada no protocolo de automação BACnet.

1.4 Apresentação do manuscrito

No Capítulo 2, é realizado uma fundamentação das bases do projeto, apresentando para isso a teoria necessária para o entendimento do trabalho desenvolvido neste projeto. Já no Capítulo 3 é proposto o acompanhamento dos materiais utilizados bem como os recursos. Continuando no mesmo, é observado como se procedeu o experimento e como foram realizadas as atividades propostas. No Capítulo 4 tem-se uma junção dos dados observados, salientando os objetivos propostos e relacionando-os com as características de projeto. Temos então por fim uma conclusão do trabalho no Capítulo 5, além de um conjunto de projetos de interesse que podem servir como objetivo para futuras pesquisas ou melhorias de trabalho.

Capítulo 2

Fundamentação

Neste Capítulo é realizada uma descrição teórica sobres os artigos tratados e os fundamentos do projeto.

2.1 Sistemas NCS

Sistemas de controle em rede, ou NCS, são sistemas onde o loop de controle é fechado por meio de uma rede. O que realmente define a NCS é que a troca de dados entre o controle e a planta é realizada por meio de pacotes de dados dentro de uma rede, que pode ter diversos protocolos e meios físicos.

Ao tratar-se de sistemas de controle em rede deve-se ter em mente como o projeto da rede e do sistema está estruturado, antes mesmo de se avaliar as opções de controle disponíveis. Nesse sentido, será avaliado uma situação real onde há a utilização de sistemas de controle em rede.[23, 24]

Dentre as diversas áreas de uso de NCS, tem-se que algumas se destacam por sua utilidade, praticidade, precisão ou eficiência. Por exemplo, quando há uma operação remota de um robô de resgate ou identificação, onde a estrutura física do local é insegura ou instável para a ação humana direta, ou em projetos de telecirurgias, como mostrado na Figura 2.1, onde utiliza-se comumente NCS. No caso da automação, temos um uso crescente de sistemas em rede, por tanto, os estudos na área de NCS voltados para automação estão ganhando destaque crescente no meio acadêmico.

De maneira geral, observa-se NCS com projetos baseados em cabeamento estruturado ou em sistemas wireless (WNCS - Wireless Networked Control System). O sistema pode ser baseado em uma rede privada e restrita, ou até mesmo por meio da internet, o que garante um sistema com possibilidades de implementação e projeto ainda mais versátil

Foi proposto então a ideia de se controlar em rede de automação desenvolvida para atender a área HVAC da automação predial. Numa situação real, o projeto em rede deve abranger um estudo específico de topologia e distribuição dos módulos de sensoriamento e atuação, tal que se possa identificar possíveis perturbações e atrasos associados, provocados por diversas alterações da arquitetura, ambiente disposto e até mesmo por interferência humana[25].

Com a ideia de custo e a problemática da eficiência em pauta, conclui-se que a utilização de

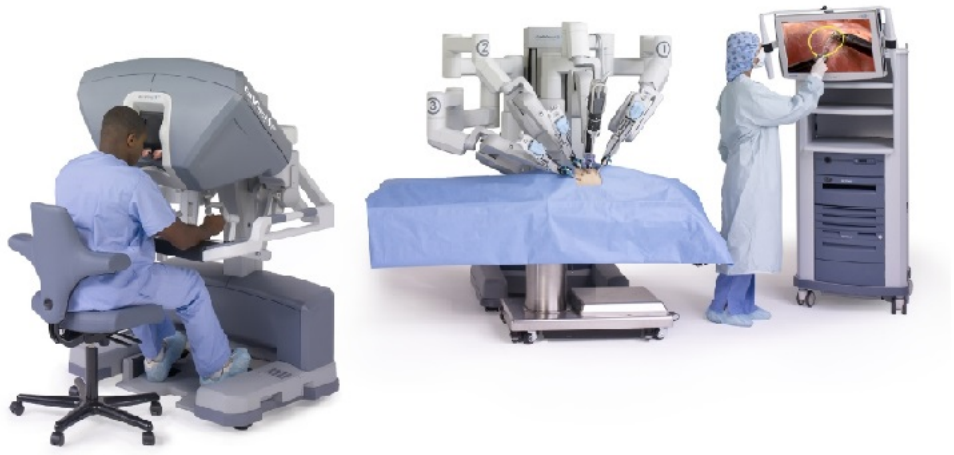

Figura 2.1: Exemplo de sistema em rede - Telecirurgia[2].

uma WNCS seria mais adequada, tanto no quesito adaptação do modelo, na capacidade de se reconfigurar o projeto quanto também na possibilidade de se expandir ou modularizar. fonte

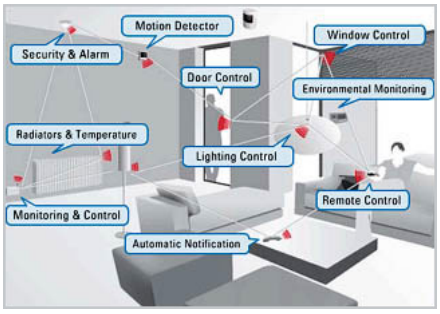

Figura 2.2: Exemplo de sistema de automação em rede - Automação residencial[3].

2.2 Modelos de Rede

Tratando-se de redes de comunicação, pode-se verificar uma vasta possibilidade de projetos. Entretanto, deve-se atentar para as características essenciais que definem o sistema e procurar garantir o cumprimento desses pré-requisitos.

O projeto visa a criação de uma rede a qual se pudesse avaliar e estudar uma NCS com facilidade de configuração, flexibilidade, possibilidade de reconfiguração diante da topologia do ambiente e custos baixos (tanto de energia quanto de implementação).

Assim, com os pré-requisitos de rede formulados, observou-se que um sistema de rede cabeado seria de certa forma uma solução dispendiosa no sentido de flexibilidade e reconfiguração da rede, nesse caso foi escolhida uma solução sem fio para atender ao objetivo deste trabalho, que é determinante para garantir flexibilidade e reconfigurabilidade diante de mudanças em uma topologia.

A utilização de redes sem fio na automação tem sido vista com bons olhos pelo mercado[26],

tanto pela relação de custo-benefício, quanto pela capacidade de se alterar sua estrutura de rede, modificando posicionamento de nós e adicionando ou retirando os mesmos de maneira mais simplificada, e com menor custo agregado. Além disso, a rede wireless facilita a configuração de projetos com características mais agressivas, tais como locais de difíceis acessos, ou custo de cabeamento elevado, e sistemas caracterizados pela mutabilidade. Pode-se verificar um exemplo de rede sem fio na Figura2.3.

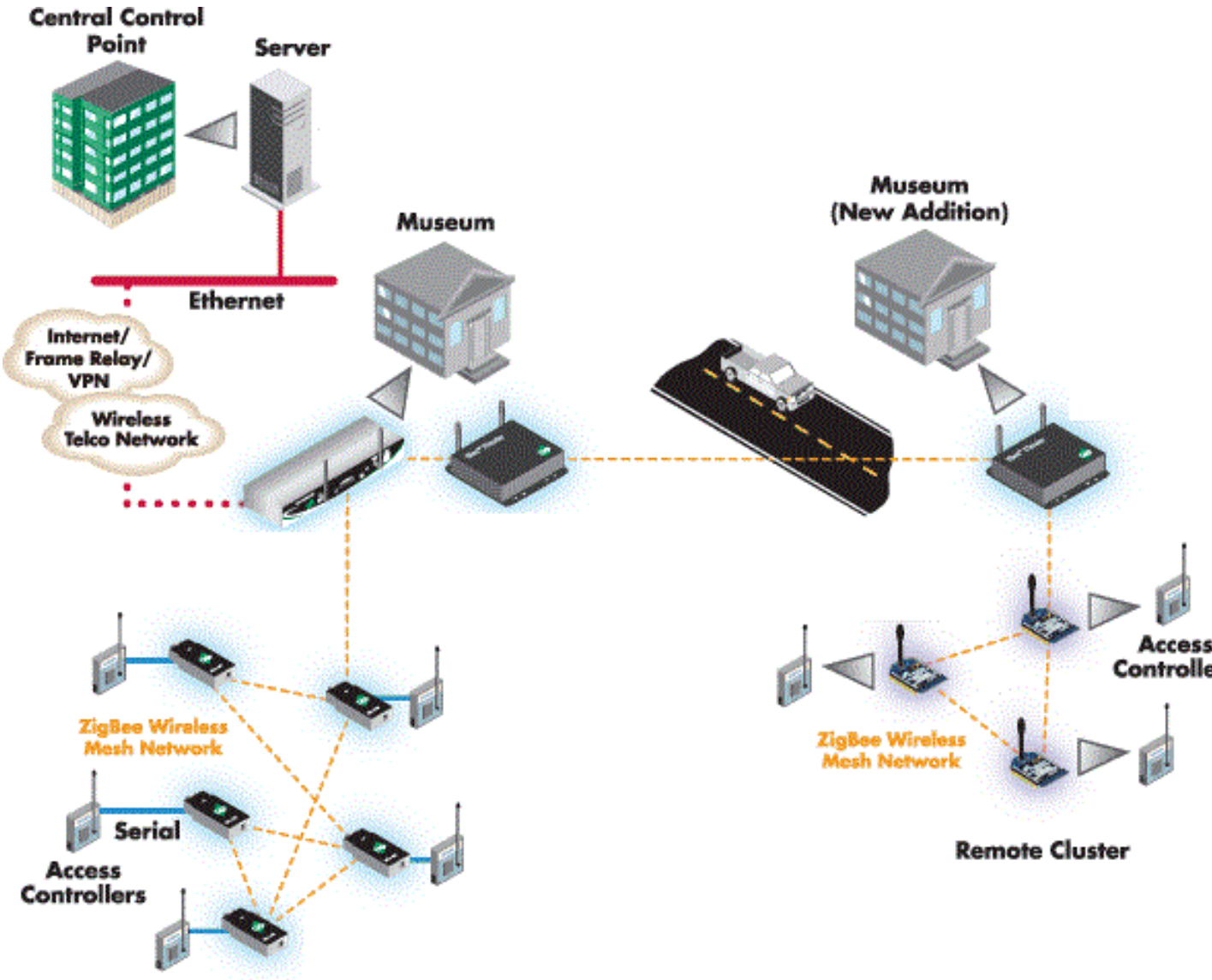

Figura 2.3: Modelo de sistema Wireless (adaptação de [4]).

2.2.1 Zigbee

O protocolo ZigBee é um padrão de rede sem fio que se apresenta como uma solução viável para o modelo, pois caracteriza-se por atender aos requisitos de maneira geral[27].

Ele é apoiado em algumas camadas do modelo de referência da organização internacional para

padronização chamado OSI [28], E tem como princípios o baixo custo em geral. É comparável com a tecnologia bluetooth ou até mesmo redes wi-fi mais conhecidas.

Uma das vantagens do ZigBee é o seu funcionamento. Os nós difundidos pela rede são de certa forma independentes, tal que o sistema se torna menos suscetível a problemas de quebra de comunicação, já que dentro da rede, quando um nó perde a comunicação, o sistema passa a se comunicar por outros nós, permitindo assim uma segurança maior da rede.

Em geral, o ZigBee possui a vantagem de adicionar nós e expandir a rede de tal forma que a flexibilidade da rede seja grande, não somente pela alteração de posicionamento dos nós, mas também pela capacidade de se expandir, limitada apenas pela configuração dos módulos. Além disso, a rede torna-se muito simples de se projetar e inserir módulos graças ao fato de que os nós entram na rede com um novo endereço de maneira automática, facilitando a instalação e configuração do sistema[27].

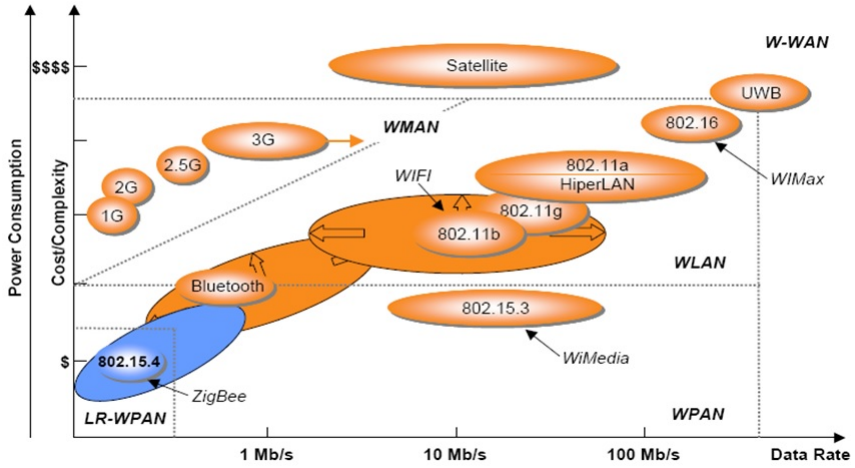

Figura 2.4: Consumo de energia/Custo x Taxa de transmissão[5].

O ZigBee, entretanto possui certas desvantagens em relação a outros padrões de rede como o Bluetooth, wi-fi, 3g por exemplo. De maneira geral, o ZigBee apresenta uma taxa de transferência de dados reduzida, tornando-o menos apropriado a aplicações com necessidades de alta taxa de comunicação.

Outra desvantagem seria a distância capaz de se comunicar. Devido ao baixo consumo o micro controlador Atmega1281R (zigbit) utilizado neste trabalho apresenta uma área de transmissão reduzida (até 70 metros) em comparação com outros tipos de padrão, como por exemplo o wi-fi que tem capacidade de transmissão de aproximadamente 100 metros.[13]

Considerando que o trabalho proposto foi desenvolvido para um sistema de controle de temperatura com baixa taxa de transmissão de dados, temos que as desvantagens apresentadas não são significantes para o funcionamento do sistema implementado. Adiante será explicado de maneira mais aprofundada o equipamento utilizado para implementação da rede ZigBee.

2.3 Maquete de automação predial

Para a implementação de ambiente similar ao encontrado em prédios, foi utilizado neste projeto uma maquete de automação de madeira, onde o atuador utilizado para fornecer carga térmica ao ambiente é um secador de cabelo, a maquete de automação utilizada está ilustrada na Figura2.5. A mesma apresenta estruturas internas (janelas e portas) capazes de simular uma perturbação ou variação no atraso da propagação da carga térmica no sistema, estes atrasos são os principais objetos de estudo neste trabalho.

Figura 2.5: Maquete de madeira com mútiplos acessos.

O uso do secador de cabelo como fonte de calor, reitera a proximidade do projeto à área de automação predial HVAC. Observou-se aqui que a humidade e a transferência de calor externa à maquete são fatores importantes, entretanto, não serão levados em consideração matematicamente em prol da simplificação da modelagem do sistema.

2.4 Identificação

O ambiente de experimentação da maquete de automação predial permite uma análise simplificada do problema proposto. Optou-se pelo uso de uma fonte de calor para quesitos de simulação, onde o atuador responsável pela fonte térmica foi um secador de cabelo Figura 2.6, com potência de $1200W$. Foi utilizada uma sala para representar o ponto de acesso da fonte térmica ao sistema. Na sala anexa, foi disposto o sensor de temperatura e o nó periférico responsável pelo retorno da comunicação. Pode-se observar claramente na Figura 2.7 a disposição do sensor e do atuador dentro do sistema.

Para que haja troca de calor entre os ambientes deve-se considerar a propagação térmica entre o ambiente da fonte, e o do sensor de temperatura. É possível compreender que um processo térmico

Figura 2.6: Secador utilizado como fonte térmica.

é característicamente lento, possui diversas variações de seus parâmetros e em geral apresenta multiplas entradas e saídas associadas. Assim, foi-se necessário a utilização de uma simplificação para representar o processo térmico e possibilitar a identificação da planta. O processo de identificação apresentado neste trabalho foi baseado no descrito em [6].

Tem-se na Figura 2.8 uma simplificação da propagação térmica observada em um sistema com uma parede, dessa forma:

$q_x \rightarrow Vetor\ de\ transferência\ de\ calor\ ao\ longo\ da\ parede$;

$x \rightarrow Posição\ ao\ longo\ da\ parede\ com\ valor\ máximo\ L$;

$T_{\infty,i} \rightarrow Temperatura\ estacionária\ na\ região\ i$;

$T_{S,i} \rightarrow Temperatura\ superficial\ da\ parede$;

De maneira geral, a propagação é realizada por convecção em cada um dos fluidos de cada lado da parede, e pela condução de calor pelo meio físico da parede. Baseado na Lei de Fourier para transmissão do calor por meio da condução tem-se:

$$q = \frac{kA}{L}(T_{S,1} - T_{S,2}) \tag{2.1}$$

Onde:

$k \rightarrow Constante\ de\ condutividade\ térmica$;

$A \rightarrow Área\ da\ superfície\ da\ parede$;

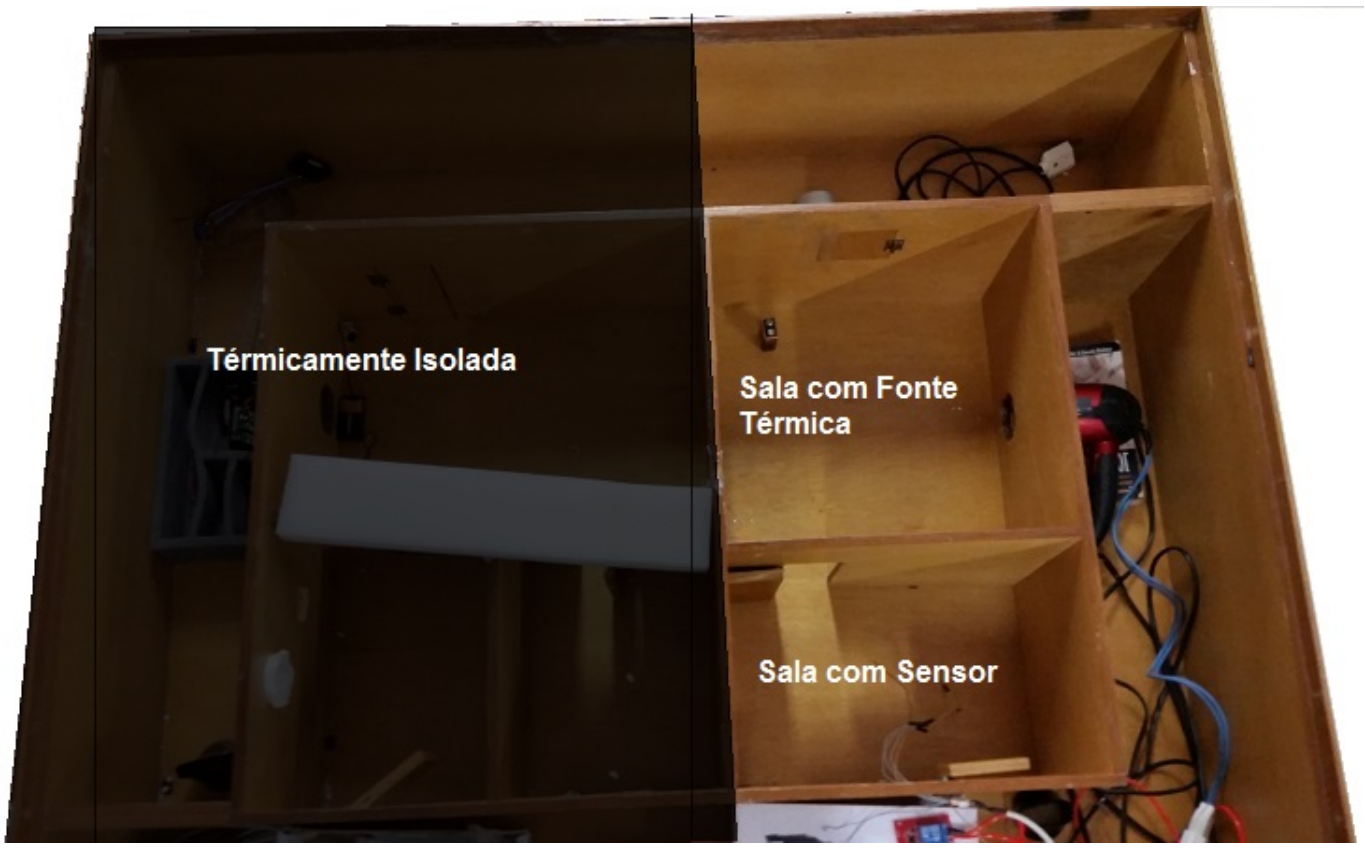

Figura 2.7: Vista superior da Maquete de Automação Predial.

$k \rightarrow Espessura\,da\,parede.$

Observa-se que o sistema é de forma geral independente do posicionamento x ao longo da parede. Nesse sentido, tem-se como uma possível simplificação do sistema uma associação do modelo térmico com um modelo elétrico. A relação entre eles é aproximada por uma capacitância interna e duas resistências, e a esse modelo de primeira ordem é dado o nome de $2R1C$. [Fonte modelo térmico]

Similarmente a um sistema elétrico, tem-se uma resistência térmica associada ao processo de condução e convecção. Essa associação se dar por meio da analogia entre resistência térmica e resistência elétrica. Assim como a resistência elétrica onde é visto uma relação entre a diferença de potêncial e o fluxo de carga elétrica, tem-se a resitência térmica com uma relação direta entre diferênça de temperatura e fluxo de calor. Tal como observado:

$$Condução \;\rightarrow\; R_{cond} = \frac{T_{s,1-}T_{s,2}}{q_x} = \frac{L}{kA}$$

$$Convecção \;\rightarrow\; R_{conv} = \frac{T_{s-}T_{\infty}}{q} = \frac{1}{hA}$$

Sendo h um coeficiente de proporcionalidade, que dependerá da superfície exposta, do calor específico do corpo e também é função de características do meio ambiente.

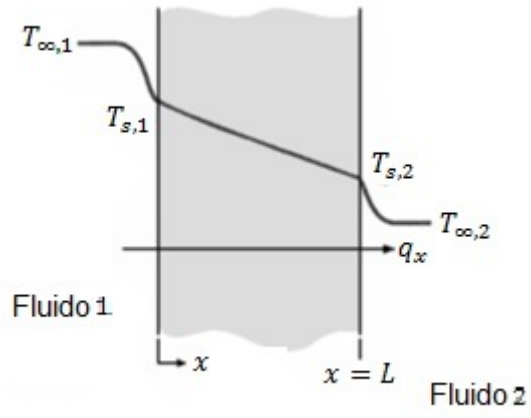

Figura 2.8: Representações das trocas de calor por meio da parede.[6]

Assim, considera-se duas resistências por convecção e uma por condução em série, tal que tem-se um sistema de grau um do tipo:

$$G_t = \frac{1}{sRC + 1} \tag{2.2}$$

Onde a constante de tempo do sistema é $\tau = RC$, e representa o período para que se alcance o equilíbrio térmico entre as duas regiões analisadas no processo. Na Figura 2.9 é visto que o processo térimco avaliado pode ser simplificado em um circuito elétrico característico, onde os parâmetros R e $C_v$são diretamente relacionados as características do sistema e da planta real.

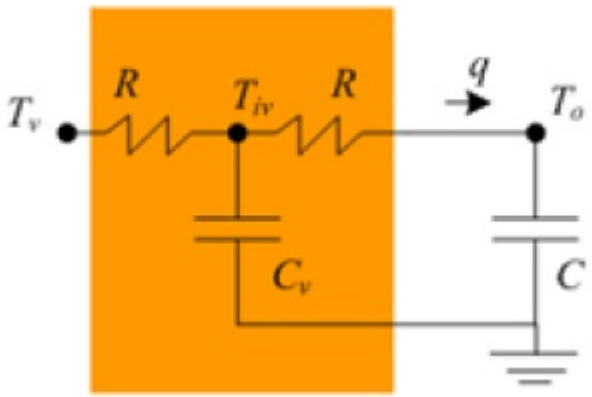

Figura 2.9: Representação do modelo da parede adaptado de [6].

2.5 Controle em rede com Transformada de *Scattering*

Um dos problemas associados ao uso da arquitetura NCS é a instabilidade do atraso do tempo na rede. Esse atraso variado promove uma deterioração do desempenho do controle e pode provocar até mesmo a instabilidade.

Para que esse tipo de problema seja solucionado existem alguma medidas possíveis, tal como o uso de redes específicamente desenvolvidas para o uso de NCS. Essa modelagem de rede especificamente desenvolvida pra essa operação permite uma garantia em qualquer caso de pelo menos uma variação de atraso dentro de uma faixa de operação.

Entretanto, quando o sistema é associado a uma rede generalizada, isto é, redes como Ethernet, o atraso no tempo e as perdas de pacotes de dados são caracterizados por serem imprevisíveis e por não poderem garantir nenhuma faixa de operação ou valores máximos.

Metodologias de controle de atraso constante são divididas em duas categorias: dependente do atraso e independênte do atraso, de acordo com quando uma faixa de valor de atraso de tempo é necessário para garantir estabilidade ou não,[7].

Algumas das metodologias de controle dependente do atraso incluem o PID especificamente ajustado e a *Smith Predictor,* esta que possui um bom desempenho, apesar de necessitar da planta inteira e da constante nominal de atraso.

Metodologias de controle independentes do atraso são normalmente baseadas no Teorema de Baixo Ganho [7], que necessita que o ganho de malha aberta em todas as frequências sejam menores que um. O problema associado a essa modelagem é a dificuldade do uso em aplicações reais, tal que integradores livres na função de transferência de malha aberta são impossibilitados, o que nesse caso promove um erro estacionário inevitável.

Uma outra alternativa de modelagem baseada em uma metodologia de controle independente de atraso é a Transformada de Scattering.

Inicialmente desenvolvida da teoria clássica de redes com o intuito de descrever o fluxo de energia em redes [7], a Transformada de Scattering se baseia na passividade da planta e do controlador, e que por tanto promove uma restrição para aplicações em geral. Assim, um sistema modelado por Scattering não poderia apresentar grau maior que dois, onde se caracterizaria a não passividade.

Observamos em [7] e mais a frente em [24] que a Tranformada passou a ser modelada e utilizada para descrever sistermas não passivos e assim, possibilitar o controle em redes mais generalizadas. Essa metodologia não necessita do conhecimento do atraso nominal da rede para implementação do controlador e se mostra menos influênciada pelas variações de atraso do sistema em rede.

Em um sistema simplificado do tipo SISO (*Single Input and Single Output*) a Transformada de Scattering age nas saidas e entradas do controlador e da planta, assim como podemos observar na Figura 2.10.

Considerando agora a Figura 2.10, observa-se:

- Y_c é a saída do controlador;

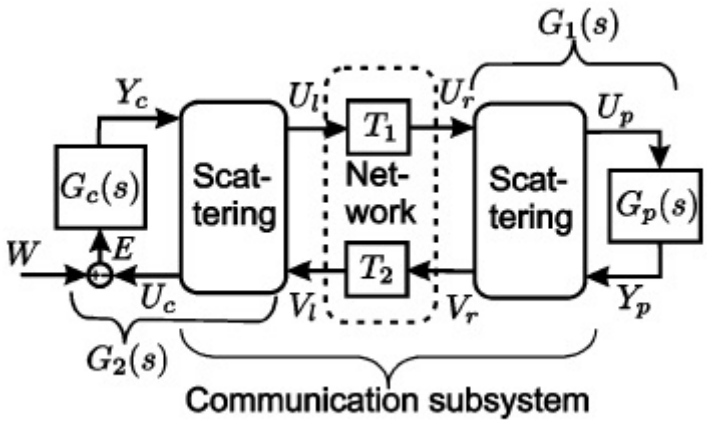

Figura 2.10: Transformada de Scatterind aplicada a uma NCS[7].

- U_cé a realimentação do controlador;
- W é a entrada de referência;
- E é o erro;
- U_p é a entrada planta;
- Y_p é saída da planta.

De maneira generalizada a Transformada de Scattering se acopla nos sistema entre os contatos da rede e do controlador e da rede e da planta, assim utilizando a entrada e saída do sistema para compor uma entrada e saída da própria transformada apra a rede. A relação generalizada entre entrada e saída com as entradas e saídas da Tranformada de Scattering podem ser vistas a seguir:

$$\mathbf{U_s} = \frac{\sqrt{2}}{2}\left(\sqrt{b}U + \frac{1}{\sqrt{b}}Y\right) \; and \; \mathbf{V_s} = \frac{\sqrt{2}}{2}\left(-\sqrt{b}U + \frac{1}{\sqrt{b}}Y\right) \tag{2.3}$$

Sendo$b > 0$, temos um sistema do tipo:

$$\mathbf{G_s(s)} = \frac{V_s(s)}{U_s(s)} = \frac{G(s) - b}{G(s) + b} \tag{2.4}$$

A partir da Figura 2.10 e da equação 2.4, tem-se os subsistemas:

$$\mathbf{G_1(s)} = \frac{V_r(s)}{U_r(s)} = \frac{G_p(s) - b}{G_p(s) + b} \; e \; \mathbf{G_2(s)} = \frac{V_l(s)}{U_l(s)} = \frac{G_c(s) - b}{G_c(s) + b} \tag{2.5}$$

Onde:

- V_r e U_r são a saída e a entrada do lado direito da *Transformada de Scattering* (em contato com a rede), respectivamente;

- V_l e U_l são a saída e a entrada do lado esquerda da *Transformada de Scattering* (em contato com a rede), respectivamente;
- G_p e G_c representam, respectivamente, a planta e o controlador.

Assim, observa-se que os atrasos T_1e T_2 ocorrem em sentidos diferentes, caracterizando dessa maneira a relação entre as variáveis da direita e da esquerda de acordo com:

$$U_r(s) = U_l(s)e^{-j\omega T_1} \tag{2.6}$$

$$V_l(s) = V_r(s)e^{-j\omega T_2} \tag{2.7}$$

Se existe um $b > 0$ tal que 2.8 é real e positivo, então o sistema de manlha fechada é estável independente do atraso no tempo[7] (A prova foi omitida por questões de simplificação dos cálculos).

$$\mathbf{K(s)} = \frac{1}{b}\frac{b^2G_c + G_p}{1 + G_cG_p} \tag{2.8}$$

Na Figura 2.11 pode-se observar o esquemático que relaciona a arquitetura característica da *Tranformada de Scattering* aplicada a uma arquitetura NCS dentro do escopo do pro apresentado. É possível verificar cada elemento dos sistema tendo uma relação direta com o que foi apresentado na Figura 2.10.

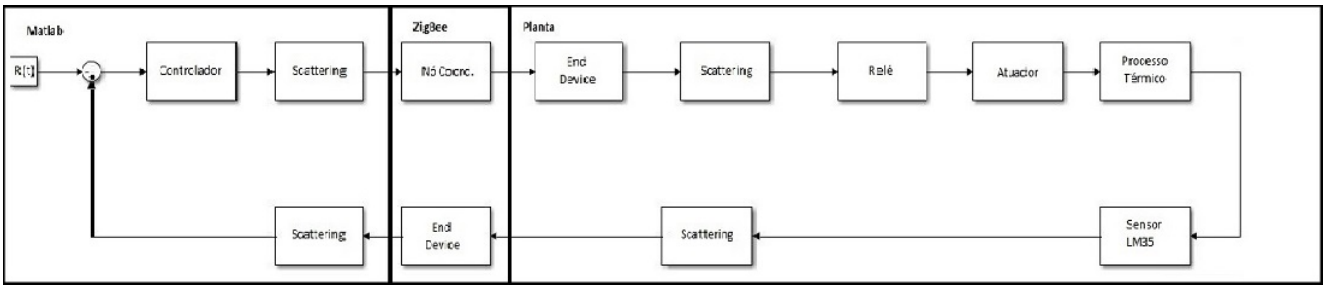

Figura 2.11: Esquemático do projeto em rede ZigBee e com Transformada de Scattering.

2.6 Integração de protocolos ZigBee e BACnet

Um protocolo de comunicação é o código que determina um padrão de linguagem para uma determinada comunicação. O protocolo em si pode ser um conjunto de regras e definições acerca de um determinado modelo de mensagem que rege a sintaxe, semântica e sincronização da comunicação.

Cada protocolo tem sua especificidade, e pode ser implementado por meio de hardware, software ou a junção dos dois. Os protocolos são imprescindíveis para os sistemas de rede de comunicação,

pois assim como uma determinada linguagem determina a comunicação em uma mensagem falada ou escrita, um protocolo define uma transmissão de dados e a capacidade de comunicação entre o receptor e o emissor. A violação do protocolo de comunicação pode ocasionar erros na comunicação ou até mesmo impossibilitá-la.

Não existe um protocolo correto ou errado, entretanto cada protocolo possui um conjunto de características que se adaptam de maneira especifica a cada problema. Assim, é necessária uma análise profunda do problema ou do sistema de comunicação para que o protocolo seja selecionado de forma adequada.

Diversos protocolos de comunicação foram criados ao longo dos anos devido as necessidades do mercado e com o tempo vêm recebendo constantes apoios e recursos para se desenvolverem. Alguns desses protocolos são MODBUS, FIELDBUS FOUNDATION, PROFIBUS, BACnet, entre outros.

Por tratar-se de um conjunto de regras ou definições, protocolos diferentes normalmente possuem diferenças entre seus códigos, mesmo sendo eles destinados à mesma área de uso, podem possuir certas especificações semelhantes. Por exemplo, as características de handshaking ou o meio como iniciar ou finalizar uma mensagem podem ser iguais para dois protocolos, enquanto detecção de erro ou perda de pacote se dá de maneira completamente diferente.[29, 28]

Nesse sentido, ao trabalhar-se em um sistema real, muitas vezes temos a necessidade de utilizar equipamento com diferentes protocolos e a partir desse ponto é que depara-se com o problema da comunicação entre as partes.

É necessário então um sistema ou um procedimento ao qual possa interpretar a mensagem em um tipo de protocolo e traduzi-la para o outro protocolo em questão. Dessa forma, existe a possibilidade de mais de um protocolo estar presente no mesmo sistema ou na mesma rede.

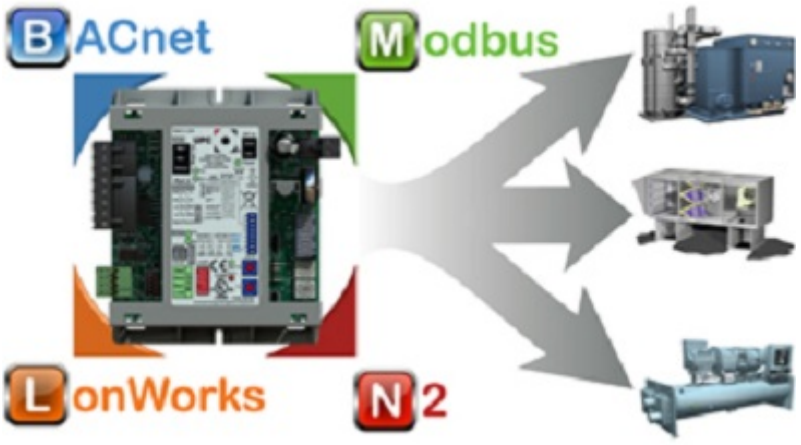

Figura 2.12: Equipamento que permite integração de protocolos[8].

A integração de protocolos permite que mais de um protocolo presente na mesma rede se comunique de forma contínua. Há entretanto limitações para esse tipo de integração, já que para que se possa existir uma integração completa entre dois protocolos, os dois devem ser abertos ao mercado.

Ou seja, um protocolo caracterizado como fechado, não pode ser de maneira geral integrado sem que haja uma abertura mínima de seus parâmetros para o responsável pela integração[28].

Nesse projeto, buscou-se também integrar a rede sem fio caracterizada pelo protocolo aberto ZigBee, com o protocolo BACnet, amplamente utilizado e difundido no mercado de automação predial e abrange grande parte dos padrões dos equipamentos.

Capítulo 3

Desenvolvimento

Neste capítulo, será feita uma análise dos materiais utilizados, bem como suas características e o procedimento do projeto como um todo.

3.1 Cenário de Testes

3.1.1 Zigbee, Zigbit e Bitcloud

Hoje em dia os sistemas de automação estão cada vez mais competitivos e nesse sentido um projeto com baixo custo energético e de implementação ganha bastante visibilidade no mercado.

Baseado no padrão IEEE 802.15.4 temos um protocolo capaz de garantir grande parte das especificações comumente vistas em sistemas de automação predial.

Enquanto o acesso físico de dados e o recebimento e envio de pacotes é baseado no padrão IEEE, a camada de rede e de aplicação, responsável pelo controle de entrada e saída de dispositivos da rede e pela interface do usuário, respectivamente, são responsabilidades do padrão ZigBee.[28, 29]

Dentro da organização de rede ZigBee podemos destacar as topologias possíveis para se ter um sistema em rede. Existem então três topologias observadas: Redes em estrela, em Árvore ou em Malha[28].

Cada uma das topologias apresenta certas características e certas aplicabilidades. Uma rede tipo estrela tem um nó central, que está ligado a todos os outros nós na rede. Todas as mensagens viajam via o nó central. Já a rede em árvore, ou tree, tem um nó superior com uma estrutura *branch* / folha. Para chegar ao seu destino, uma mensagem viaja em direção ao topo árvore e, em seguida, para baixo da árvore até o nó destino. Uma rede de malha ou mesh tem uma estrutura de árvore em que algumas folhas estão diretamente ligadas. As mensagens podem viajar por toda a árvore, quando uma rota adequada está disponível e também podem tomar rotas alternativas caso a comunicação com um determinado nó seja interrompida.

Em termos de rede, o protocolo ZigBee é caracterizado pela presença de três equipamentos específicos ou três tipos de nós: nó coordenador, nó roteador e End Device. Esses dispositivos desempenham um papel específico na rede cada um.

Em primeiro lugar temos os dispositivos coordenadores, que são os nós centrais da rede. Eles são responsáveis pela administração da rede de uma forma geral, selecionam o canal de frequência da rede, inicia a rede e permite que outros dispositivos se conectem a ele. Também agregam a responsabilidade de transmitir mensagens entre nós finais.

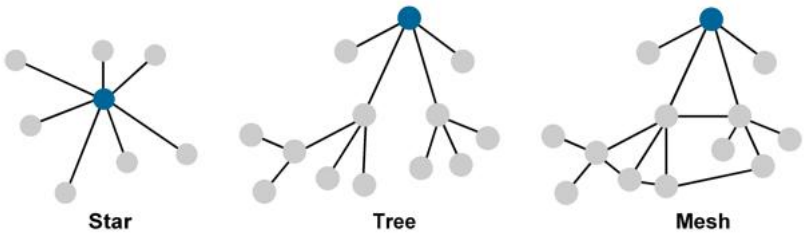

Figura 3.1: *Nós coordenadores em azul dentro das três topologias[9].*

Os dispositivos de fim de rede são os responsáveis pelo contato direto com os sistemas físico, como por exemplo atuadores que, nesse caso, são diretamente conectados aos equipamentos finais e que assim podem receber um sinal de controle por meio da rede ZigBee. Além de atuadores, outros tipos de equipamento podem ser atrelados aos dispositivos de fim de rede, tais como sensores e integradores. Enquanto não utilizados dentro da rede, os dispositivos finais podem ser colocados em modo de latência ou sleep, como explicado mais à frente.

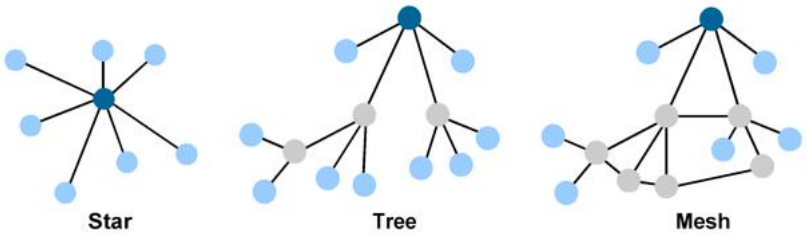

Figura 3.2: *Nós finais de rede em azul anil dentro das três topologias[9].*

Já os roteadores são dispositivos utilizados na retransmissão de dados, tal como repetidores de sinais, onde possibilitam um maior alcance físico da rede e uma maior conectividade entre os nós. Dentro das redes tipo estrela é observado que a função de roteamento das mensagens é realizada exclusivamente pelo nó coordenador.[10, 27]

Para que fosse possível utilizar uma rede ZigBee é necessário foram utilizados equipamentos específicos para o projeto. Nesse caso foram utilizados os módulos Zigbit (ATZB-24-A2/B0).

Os módulos ZigBit são módulos wireless desenvolvidos pela Atmel com baixo consumo de potência, alta sensibilidade e com frequência de processamento de 2.4GHz. Além disso, o Zigbit é um modulo de tamanho reduzido, com menos de 2 cm^2 de espaço o que garante ao produto final módulos com tamanhos reduzidos.

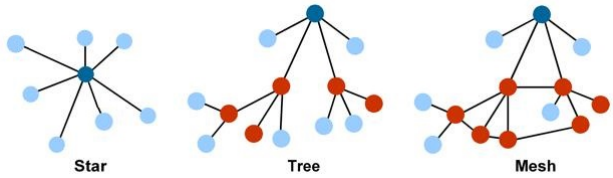

Figura 3.3: *Nós roteadores em laranja dentro das três topologias[9].*

O ZigBit possui diversas entradas e saídas para que possa ser acessado. Dentre elas destacamos as bases de IRQ (usadas para interrupção), ADC (referente a conversão analógico digital), porta UART e USART, entrada I2C (utilizada para sensores digitais), GPIO (entradas digitais), entre outras.[10]

Os equipamentos utilizam a comunicação por meio da UART que é o formato de comunicação serial padronizado. Essa padronização não deve ser apenas garantida pelos parâmetros da UART, mas também pelo padrão de comunicação ente os dispositivos e equipamentos dentro da rede de comunicação. Nesse sentido, é necessário se mantenha certas configurações como mesma banda de comunicação, mesma velocidade de comunicação, mesma velocidade de processamento, para que seja garantida uma maior confiabilidade da rede.

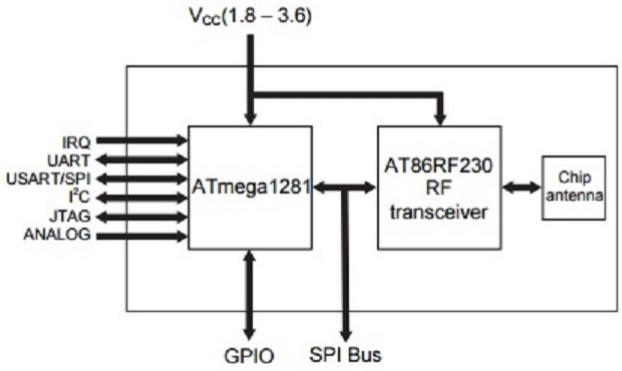

Figura 3.4: *Esquemáticos de I/O do ZigBit,[10].*

O ZigBit contém um microcontrolador Atmel ATmega1281V e um Transceiver RF Atmel AT86RF230. Possui, além disso, uma memória flash de 128KB e uma memória RAM de 8KB. Todos os ZigBits são pré-carregados com um gerenciador de inicialização (Bootloader) quando eles são vendidos como módulos, ou em unidades individuais T & R.

Pode-se ver na Figura 3.4 uma ilustração do módulo ZigBit, na mesma pode-se observar os pinos das entradas de comunicação, das entradas analógicas, e a localização da antena no módulo.

Dependendo dos requisitos de design para o usuário final, um módulo ZigBit pode funcionar como um nó sensor auto-suficiente ou pode ser emparelhado com um processador do host, o módulo de condução por meio de uma interface serial. No primeiro caso, uma aplicação pode ser utilizada

com o software BitCloud, permitindo a personalização das aplicações embarcadas através da API C do BitCloud. A seguir serão definas algumas das especificações referentes ao Zigbit.

Parameter	Minimum	Maximum
Voltage on any pin, except RESET with respect to ground	-0.5V	V_{CC} + 5V
DC current per I/O pin		40mA
DC current DVCC and DGND pins		200mA
Input RF level		+10dBm

Figura 3.5: *Máximos parâmetro dos módulos[11].*

A Figura 3.5 ilustra a tensão em qualquer um dos pinos, bem como a corrente máxima permitida em cada pino, além da potência do pino de entrada do tipo RF.

Parameter	Range	Unit
Supply voltage, V_{CC} (2)	1.8 to 3.6	V
Current consumption: RX mode	21.8	mA
Current consumption: TX mode (1)	20.8	mA
Current consumption: Radio is turned off, MCU is active 50% of the time (1)	3.5	mA
Current consumption: Power-save mode (1)	6	µA

Figura 3.6: *Entrada e saída do módulo[11].*

Na Figura 3.6 é apresentado o consumo de corrente para os modos de operação do ZigBit.

Parameter	Condition	Range	Unit
Frequency band		2.4000 to 2.4835	GHz
Numbers of channels		16	
Channel spacing		5	MHz
Transmitter output power	Adjusted in 16 steps	-17 to +3	dBm
Receiver sensitivity	PER = 1%	-101	dBm
On-air data rate		250	Kbps
TX output/ RX input nominal impedance	For balanced output	100	Ω

Figura 3.7: *Características do Transceptor[11].*

Na Figura 3.7 podemos ver alguns parâmentros da rede de comunicação gerada pelos módulos.

Parameter	Condition	Range	Unit
On-chip flash memory size		128K	Bytes
On-chip RAM size		8K	Bytes
On-chip EEPROM size		4K	Bytes
Operation frequency		8	MHz

Figura 3.8: *Características do Microcontrolador ATmega1281V[11].*

Na Figura 3.8 foram mostrados valores da quantidade de memória presente no ZigBit.

Parameter	Condition	Range	Unit
UART maximum baud rate		38.4	Kbps
ADC resolution/ conversion time	In single conversion mode	10/200	Bits/µs
ADC input resistance		>1	MΩ
ADC reference voltage (V_{REF})		1.0 to V_{CC} -3	V
ADC input voltage		0 - V_{REF}	V
I^2C maximum clock		400	kHz
GPIO output voltage (high/low)	-10/ 5mA	2.3/ 0.5	V
Real time oscillator frequency		32.768	kHz

Figura 3.9: *Características de configuração do módulo (I/O)[11].*

Na figura 3.10 foi ilustrado algumas informações dos módulos I/O do ZigBit.

Foi observado que para o uso de módulos ZigBits é necessário o uso de um hardware adaptado que promova a utilização de suas portas e o uso de seu processamento de maneira eficiente e mantendo os benefícios de sua arquitetura, tais como baixo consumo, por exemplo. Nesse sentido, foram desenvolvidas placas ZigBit Channel V1.0, placas para adaptação do módulo Zigbit, utilizada tanto para os dispositivos coordenadores quanto para os finais. No projeto desenvolvido elaborou-se uma rede com um nó coordenador e um nó End Device, onde o nó coordenador é montado na placa com USB e o nó End Device é montado no modulo com alimentação externa de 3V acoplada, sendo essa alimentação feita por bateria composta de duas pilhas 1,5V. Duas diferenças a se destacar são: a placa voltada para os nós coordenadores possuem também a característica de gravadoras e são caracterizadas pela saída USB, diferentemente dos modelos de dispositivos finais que apenas possuem uma bateria anexa.

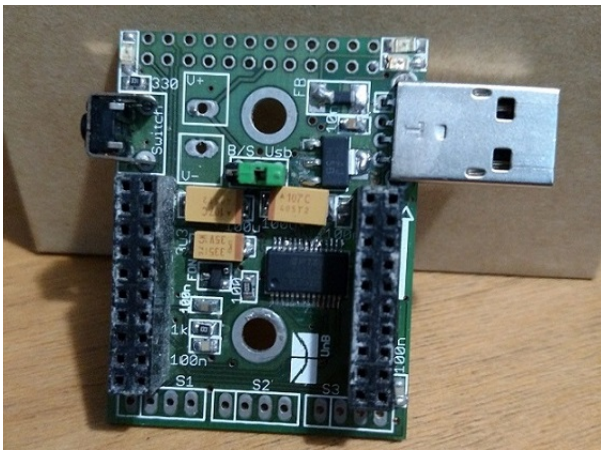

Figura 3.10: *Placa UsB do nó coordenador/gravadora.*

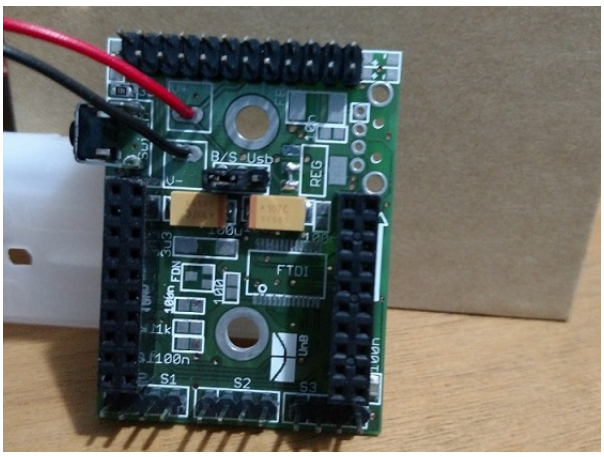

Figura 3.11: *Placa do End Device com fonte bateira.*

A placa da Figura 3.10 é utilizada para o coordenador/gravação. A placa possui um circuito de conexão USB tal que pode se conectar diretamente ao computador permitindo configurações, downloads e uploads de programação e transmissão de controle por meio da transmissão serial entre computador e a placa. A placa possui um regulador de tensão de 3,3 volts que garante a alimentação adequada do sistema, além de um botão de reset que é utilizado para gravação e reestruturação da rede formada. Já aplaca da Figura 3.11 é utilizada para dispositivos finais de rede, que necessitam apenas comandar um atuador ou transmitir uma informação de um sensor. Assim, temos que a placa é voltada para esse tipo de características de transmissão e para tanto possui uma fonte de alimentação baseada numa bateira, com a presença de um regulador de tensão de 3,3 volts, assim como a placa USB, e possui um sensor de bateria, não utilizado nesse projeto. Essas placas foram projetadas e desenvolvidas pelos Engenheiros Felipe Brandão e Vinícius Galvão. E seu esquemático será apresentado na sessão de anexos destes relatório. As placas que contém os módulos Zigbit são placas superiores (BreaKouts), caracterizadas por um conjunto de componentes passivos. Esse modelo de placa, apresentado na Figura 3.12 foi desenvolvido pelos alunos da Universidade de Brasília Felipe B. Cavalcanti e Vinícius Galvão atuais detentores dos direitos de fabricação.

Figura 3.12: *Placa Zigbit Breakout.*

Para implementar a rede wireless, as funções de leitura de sensores, o envio automático de

mensagens e o protocolo de BACnet de comunicação, Foi necessário implementar um software nos módulos ZigBit, tanto no contexto das interfaces, quanto nas aplicações em geral, e no sistema de inicialização e configuração.

Para que cada módulo funcione de acordo com seu objetivo central é necessário que ele receba a programação específica que define os processos dos módulos caracterizando-os como dispositivos coordenadores, roteadores e finais de rede. Para tanto, são disponibilizadas bibliotecas diversas (APIs), pela MeshNetics, que promovem o estudo e o desenvolvimento de aplicações utilizadas no sistema.

Uma das bibliotecas oferecidas se chama BitCloud. O BitCloud é um pacote de desenvolvimento que contém implementações de diversas aplicações e recursos para construção e programação de aplicações em sistemas com protocolo ZigBee.

O BitCloud da Atmel segue o sistema de separação de camadas de rede do IEEE 802.15.4 e ZigBee. Além da pilha central, que contém a implementação do protocolo propriamente dito, a arquitetura Bitcloud contém camadas adicionais de implementação de serviços compartilhados (gerenciador de tarefas, de energia, segurança, por exemplo) e abstração de hardwares (como hardware abstraction layer - HAL ou Board support package - BSP). Apesar das API's contidas nessas camadas estarem fora da pilha central de funcionalidades, essas funções contribuem significantemente para o desenvolvimento de novas aplicações, permitindo a redução da complexidade e simplificando as integrações.[30, 12]

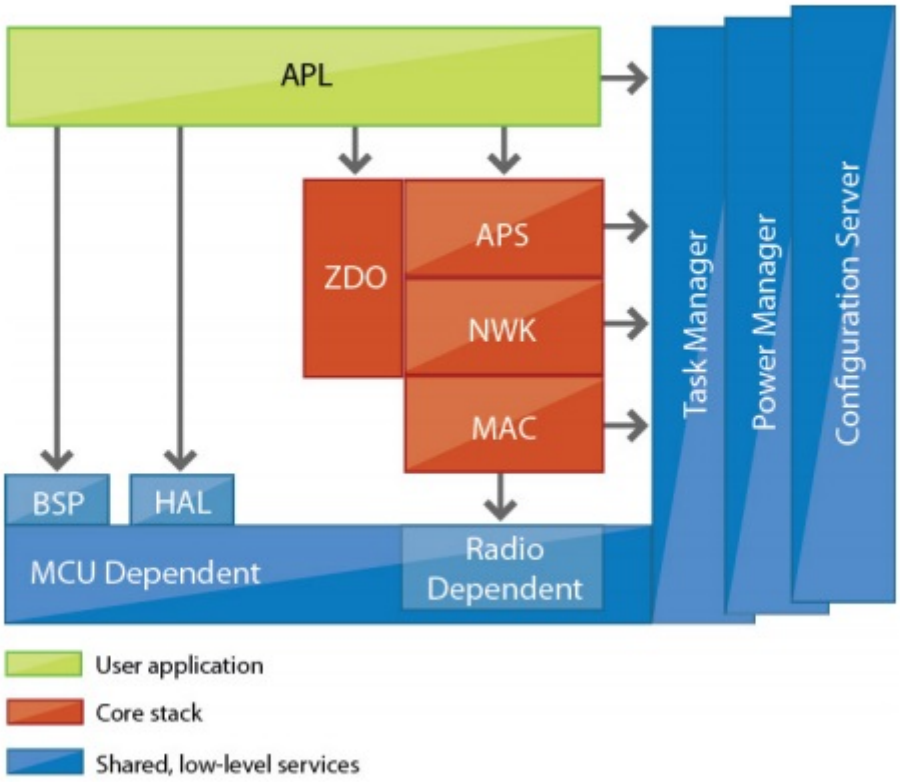

Figura 3.13: *Arquitetura da pilha de software Bitcloud,*[12].

É a partir do Bitcloud que se pode gerenciar de maneira adequada a rede e seus parâmetros. Por meio dessa pilha que podemos criar programas do sistema com características específicas tais como os programas de análise de sensores, ou por exemplo os sistemas de atuadores baseados no mesmo processo de LEDs liga-desliga. O conjunto de bibliotecas ou aplicações pré-implementadas estão descritas na Figura3.14.

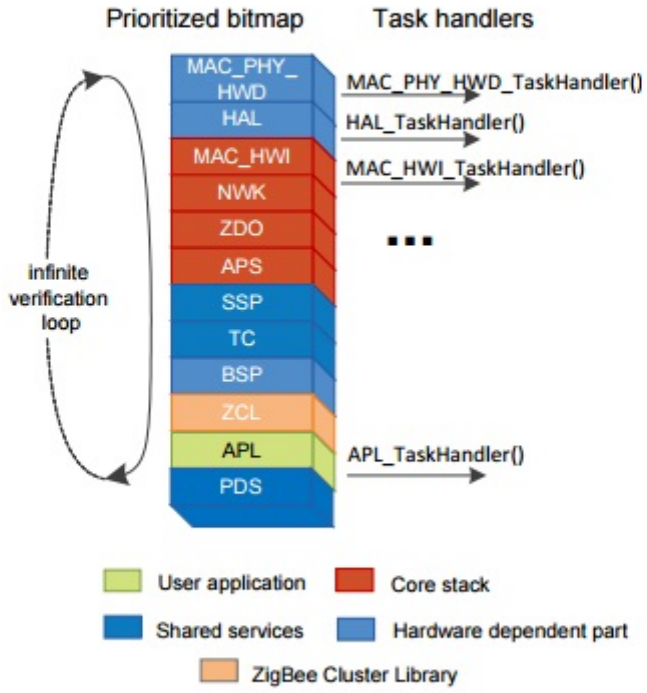

Figura 3.14: *Pilha de aplicações internas à BitCloud,*[12].

Nesse caso, temos diversas funcionalidades e adaptações para cada um dos componentes da pilha. Em termos práticos, utilizamos nesse projeto de uma série de aplicações base, dentre elas:

- MAC_PHY: Camada MAC e de acesso físico;
- HAL (*Hardware Abstraction Layer*): Inclui um conjunto completo de APIs para a utilização de recursos de hardware no módulo (*EEPROM*, *Sleep mode* e temporizadores watchdog), bem como os controladores de referência para a *design* rápido e integração suave com uma variedade de periféricos externos (IRQ, TWI, SPI, USART, e 1-*wire*);
- NWK: camada de rede;
- ZDO (*ZigBee Device Object*): possibilita correlação entre camada de rede e camada de aplicação;
- APS: biblioteca que promove suporte a sub-seções das camadas por meio de interface com o protocolo;

- BSP: inclui um conjunto completo de drivers para o gerenciamento de periféricos padrão (sensores, chips UID, *sliders* e botões) colocada sobre uma placa de desenvolvimento;
- ZCL (*ZigBee Cluster Library*): framework presente em aplicações que envolvem agrupamento de tarefas usualmente importantes para diversos tipos de aplicação. BSP ou Board support package, HAL - *hardware abstraction layer*, da supporte a camanda física.
- WSNDemo: Aplicação WSN, para aquisição de dados e análise dos mesmo, juntamente com o software WSNMonitor;
- *PersistDataServer* ou PDS: Possibilita o uso e acesso da EEPROM e garante agerenciamento.

Utilizando como base certo conjunto de aplicações possibilitadas pela API Bitcloud, foi possível implementar uma rede de comunicação consistente com um conjunto de parâmetros de configurabilidade que permitiu flexibilidade e dinâmica durante o tratamento do sistema.

Em primeiro lugar, foi utilizado um sistema baseado em uma topologia estrela, tal que o nó coordenador é o responsável pela interface de roteamento e deve estar conectado diretamente com todos os outros nós, que nesse caso seriam do tipo *End Device* de rede.

Na utilização do sistema *sleep*, foi observado que o coordenador tem uma relação de comunicação com cada um dos outros nós, tal que permite durante períodos programados, a transferência de dados de cada *End Device*, um de cada vez, organizado por uma sequência de prioridade pré-programada (como por exemplo, nó que caracterizam sinais de alarme) ou pela sequência de IDs estabelecida na entrada subsequente de cada nó na rede.

A esse período de habilitação de transmissão é denominado período de *beacon*, onde o nó coordenador permite a um determinado *End Device*, da rede, que ele transmita uma mensagem e que ocorra comunicação entre eles. Em diversos momentos o nó habilitado não promoverá uma transmissão caso não haja alteração alguma do sistema observado ou no caso de um atuador caso o nó roteador não deseje transmitir. A consequência desse sistema é uma redução significativa do custo de energia[27].

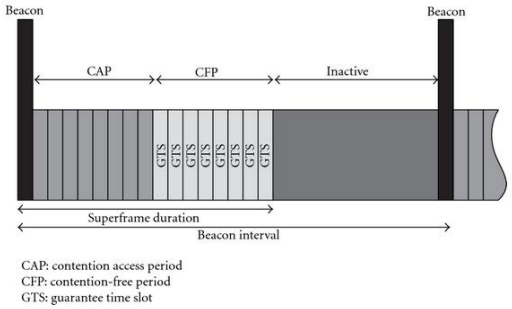

Figura 3.15: *Modelo de sistema que utiliza intervalos de Beacon[13].*

A função do nó coordenador dentro do sistema é a de mediar o conteúdo de informações enviadas pelos dispositivos *End Device* e o Matlab que é o ponto de análise de dados. A primeira

atividade estabelecida foi implementar a rede e a malha de controle no Simulink do Matlab. Após a rede estar funcionando, o coordenador se torna roteador de informações, e se torna o link direto do computador com o resto da rede (nesse caso, os dispositivos finais de rede). Em seguida, o coordenador passa a receber mensagens e a encaminhá-las para o computador para que possam ser analisadas e processadas pela malha de controle.

A implementação desse processo foi baseada em algumas aplicações previamente implementadas, inclusas na API Bitcloud. A programação do coordenador pode então ser observada na documentação em anexo.

Já a função do *End Device* caracteriza-se por garantir uma troca de informações entre ambiente e sistema de rede, ou planta e rede. Assim, no caso de um atuador, a partir do momento que o coordenador envia uma mensagem contendo um sinal de atuação, então o nó final deve emitir esse sinal de atuação. O sinal de atuação emitido pelo Coordenador é transmitido ao longo da rede e interpretada pelo End Device, que aciona o módulo relé fazendo assim com que o atuador ligue ou desligue baseado na mensagem enviada pela rede[27].

Assim, com o nó coordenador, o *End Device* também possui um código de programação especifico, o mesmo está apresentado na sessão de anexos deste trabalho.

3.1.2 Sensor de Temperatura LM35

O circuito integrado LM35 é um sensor de temperatura de precisão, fabricado pela National Semiconductor, cuja saída de tensão é linearmente proporcional à temperatura em grau Celsius. Isso permite uma vantagem com relação a outros sensores lineares de temperatura que são calibrados em Kelvin, já que o usuário não necessita realizar subtrações de grandes valores para poder fazer uma transformação da temperatura de Kelvin pra Celsius.[11]

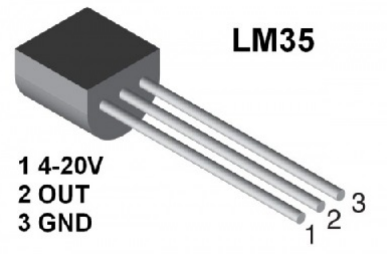

Figura 3.16: *Imagem do sensor de temperatura analógico LM35[11].*

Além disso, o LM35 não necessita de qualquer calibração externa ou *"trimming"* para fornecer com exatidão valores de temperatura tais como $\frac{1}{4}$ºC ou até mesmo $\frac{3}{4}$ºC dentro da faixa de temperatura entre -55ºC e 150ºC. O mesmo possui baixa impedância de saída, tensão linear e calibração

inerente precisa, fazendo com que o interfaceamento de leitura seja especificamente simples, barateando todo o sistema em função disso. Observou-se em nosso projeto que a saída do sensor se dava em torno de degraus de 10mV/ºC o que influenciou no código do *End Device*, pois teve-se que implementar em código uma conversão para graus Celsius.

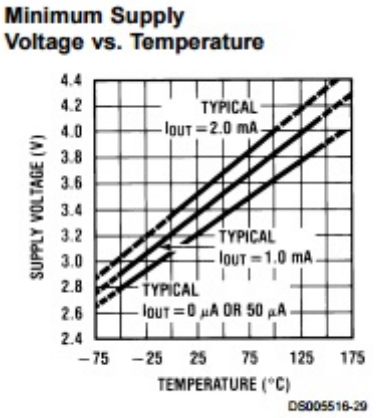

Figura 3.17: *Gráfico de mínima tensão x Temperatura,[11].*

O dado do sensor é lido por meio de uma porta de conversão ADC inclusa no sistema ZigBit. Nesse sentido podemos analisar sistemas que vão desde 8 bits de resolução até mesmo resoluções de 14 bits. Nesse projeto em específico, a resolução de leitura do dado do sensor foi escolhida para 8 bits, apesar da resolução não ser a ideal, obteve-se, para análises com longos períodos, resultados satisfatórios para a análise dos atrasos do sistema.

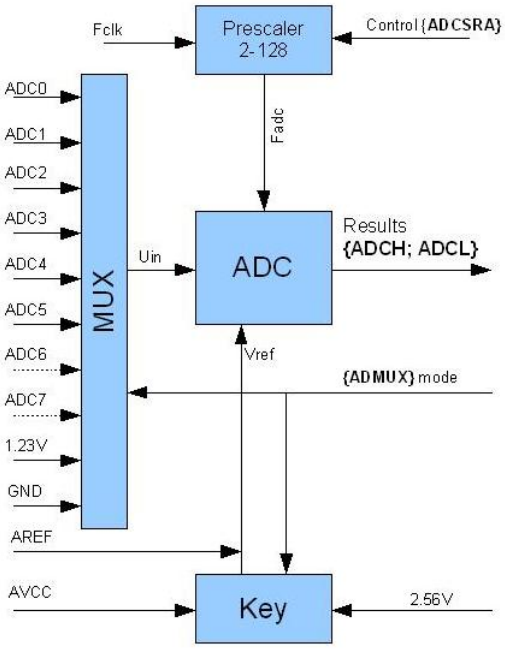

Figura 3.18: *Imagem do esquemático do ADC[10].*

3.1.3 Relés

Sistemas que utilizam microcontroladores, com fontes de alimentação de baixa tensão, necessitam da utilização de uma chave estática ou relé para acionar atuadores que sejam alimentados por tensões mais altas como 220 volts por exemplo, essa alimentação é feita através de um sinal de 3 volts do pino de saída do ZigBit, que aciona um módulo relé com oito relés, chaveando os 220 volts, fornecidos por uma fonte externa, necessários para acionar o secador para fornecer calor à planta de automação.

Existe dois modelos de relés existentes no mercado: o eletromecânico e o de estado sólido. O eletromecânico é o relé que se ativa utilizando um acionamento mecânico que acontece quando a corrente do sistema passa por uma bobina e causa uma indução magnética fechando os contatos do circuito, que inicialmente encontrava-se aberto. Por outro lado tem-se os relés de estados sólidos que constituem um conjunto mais moderno de sistema de ativação, com maior robustez no quesito problemas mecânicos e menor ruído associado.

O modelo escolhido, pela facilidade de ativação, e pelo fato de apresentar circuito mais completo e didático, foi o *SainSmart 8 Channel DC 5V Relay Module* que constitui um módulo com 8 canais de relés que podem ser utilizados de maneira rápida e simples, graças a praticidade do projeto, o mesmo encontra-se ilustrado na Figura 3.19.

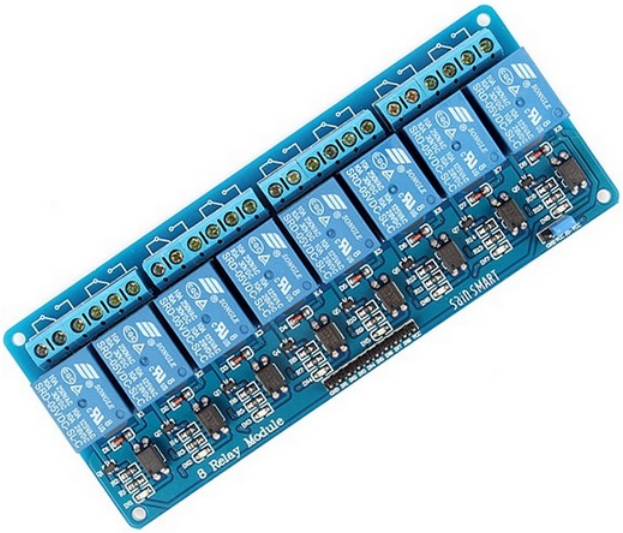

Figura 3.19: *Módulo relé de 8 canais por SainSmart.*

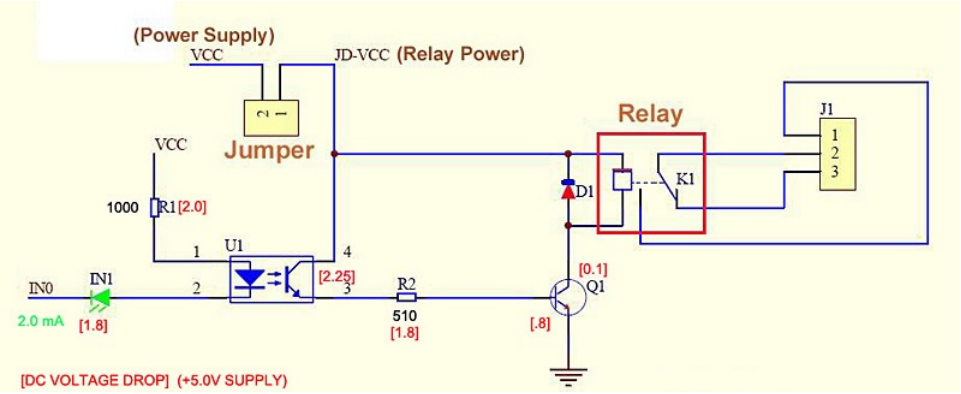

Figura 3.20: *Esquemático do circuito do relê[14].*

3.1.4 Modulação por largura de pulso ou PWM

Modulação por largura de pulso ou PWM, é uma técnica de modulação que permite a utilização de pulsos de sinais com diferentes larguras para modularizar o sinal médio de potência fornecido a um determinado equipamento elétrico.

Tem-se como base um valor de tensão que alimenta a carga e nesse sentido um sinal de controle que varia entre 1 e 0. Chamamos de DuCy ou Duty Cycle a razão entre o tempo em que o sistema passou no estado 1 (condução) e o período total do PWM (Tempo de condução + tempo de não condução). Essa relação define a tensão média fornecida ao atuador ao longo do tempo, o que consequentemente determina a potência ou energia transferida ao sistema.

Utilizamos o PWM para auxiliar no acionamento do relé. Nesse sentido, foi observado, por realização de vários testes, que o período de ajuste do PWM ideal para o trabalho foi com período de 9 segundos, pois assim o sistema modulava o comando do atuador à medida em que o setpoint de temperatura desejado fosse se aproximando do valor de temperatura medido pelo sensor.

3.2 Procedimento

3.2.1 Rede Zigbee e Processamento

Para realizar o estudo da influência dos atrasos da propagação térmica e o efeito das perturbações no comportamento dos controladores, foi montada uma rede wireless composta por 3 elementos, sendo eles o computador, o nó coordenador e o *End Device*.

Para iniciar a coleta de dados para a análise efetiva da eficiência dos controladores implementados, foi necessário primeiro achar a variável "b" ótima para o controlador scattering. Como esta variável é dependente da planta onde o controlador é implementado, este parâmetro é achado de

forma empírica, onde foram aplicados valores de atrasos e analisado o comportamento do controlador, para a excitação ao degrau, durante a execução do experimento.

Os valores de atraso utilizados para Td e Td_n, onde Td e Tdn foram tomados como sendo iguais inicialmente, foram 0.1, 0.2, 0.6 e 1, como também foram usados os valores 1.7, 1.8, 1.9, 1.95 e 1.2 para encontrar-se o parâmetro "b" ótimo para scattering.

Para encontrar o parâmetro ótimo para a scattering foram realizadas combinações dois a dois dos valores de variáveis escolhidos para que fossem realizados a maior quantidade de testes possíveis, aumentado assim o grau de confiabilidade da escolha do parâmetro. Após os testes realizados verificou-se que o parâmetro "b" ótimo é 1.95 para a maquete de automação predial.

Para validação dos estudos deste trabalho foram utilizados três controladores, e foram realizados testes com atrasos variados, onde em um primeiro momento foram testados atrasos reais (Td) de mesmo valor dos atrasos de projeto (Td_n), onde foram variados os atrasos reais para observar o comportamento dos controladores frente a um comportamento que não seja o ideal encontrado em projeto.

Ao aplicar-se a variação nos atrasos foi constado que o único controlador que conseguiu realizar um controle estável foi o scattering, pois resposta dos outros dois modelos mostrou-se ruim e mesmo instável, fazendo com que o controle perdesse a sua qualidade e capacidade de manter a temperatura no Setpoint desejado. Na sessão de dados deste trabalho serão apresentados os gráficos com os resultados obtidos nos experimentos para provar a validade do estudo realizado.

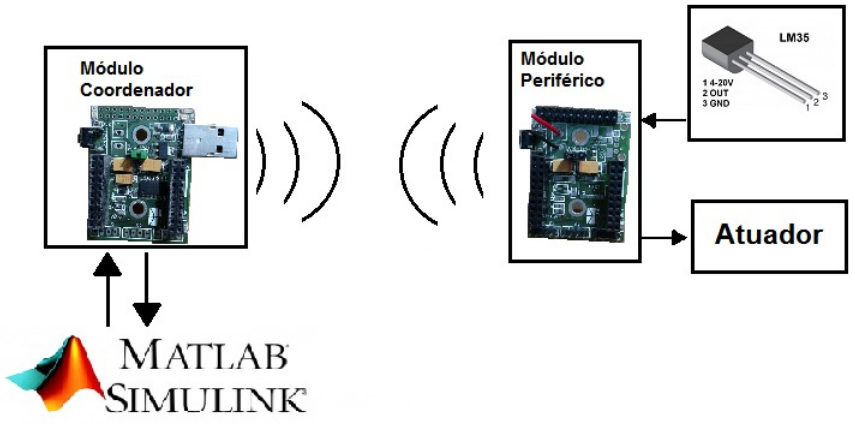

Figura 3.21: *Esquemático da Rede utilizada.*

3.2.2 Sistemas de Controle

O sistema de controle deste projeto visa comparar, de maneira abrangente, o desempenho de cada um dos três modelos implementados para a malha de controle. Para elabora-se a malha de controle com realimentação, foram utilizados os modelos de controladores a seguir[7]:

$$Scattering \;\rightarrow\; Gc(s) = \frac{1.3981(1000(s+9.9114))}{((s+9.1558)(s+1000))}$$

$$Smith\,Predictor \;\rightarrow\; Gc(s) = \frac{5(s+8.4642)}{(s+22.5354)}$$

$$Atraso \;\rightarrow\; Gc(s) = \frac{3.2649(s+4.7877)}{(s+57.7271)}$$

Na Figura 3.22 temos um organograma que demonstra a estrutura criada para cada um dos modelos. Cada bloco representado na figura possui sua própria estrutura de realimentação característica de cada modelo. A saída do sistema foi testada para cada um dos modelos utilizando como modulador o PWM.

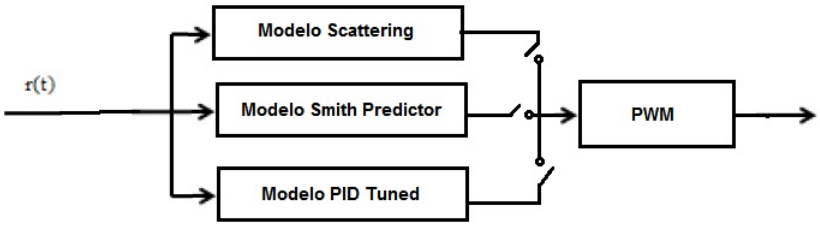

Figura 3.22: *Modelo simplificado da estrutura de controle.*

Cada um dos modelos possui uma característica específica que promove e destaca sua qualidade em determinado problema.

Tendo como base a planta característica do sistema da maquete de madeira, procurou-se implementar os modelos de forma que a comparação dos sistemas fosse justa, já que está sendo descrito um problema onde dois modelos dependentes do atraso no tempo são comparados com uma solução menos dependente desse atraso.

Assim, para que fosse possível gerar uma comparação adequada, buscou-se utilizar um modelamento baseado em um critério de medida. O critério de medida utilizado foi o ITAE (Integral Time-weighted Absolute Error), que promove uma integral do erro ponderado pelo próprio período.[31]

$$ITAE = \int \tau |e(\tau)| d\tau \tag{3.1}$$

O critério se baseia no fato de que, como o seu valor equivale ao erro total do sistema, foram criados parâmetros iniciais dentro de cada modelo para que utilizando o sistema de minimização fosse possível encontrar os parâmetros dos controladores.

De uma outra forma, podemos imaginar que se o erro total do modelo for dado pela equação 3.1,temos que considerar que o objetivo de qualquer análise e modelagem seria a minimização desse erro tal que fosse encontrado um conjunto de fatores que representassem essa função erro e que combinadas daria um resultado de mínimo valor para ITAE. Sendo possível encontrar esse valor, teve-se então uma descrição de modelo passível de ser utilizado.

Para tanto, teve-se que ter em mente que solucionar de maneira direta esse problema não é trivial, pois a proporção computacional de dados necessário seria extremamente grande. Sabendo disso, foi utilizado a ferramenta de análise de mínimos parâmetros do Matlab chamada FMINCON. Essa ferramenta permite que se entre com os dados da equação 1, os valores de período de amostragem e tempo de atraso associado, os parâmetros iniciais e condições de controle sobre os parâmetros, retornando o conjunto de parâmetros associados que minimizam a equação 3.1[7, 24].

Utilizando como base a planta da maquete, e considerando um valor nominal do atraso de tempo T associado pode-se utilizar a função FMINCON com sucesso.

Como uma estrutura base simples utilizamos um controle em avanço-atraso *(lead-lag)* tal como:

$$G_c(s) = k\frac{s+a}{s+c} \tag{3.2}$$

Os parâmetros não especificados k, a, e c, serão os parâmetros encontrados pelo método de minimização que ocorre de maneira similar pra cada um dos modelos.[7]

Para que o modelo tenha sucesso é necessário avaliar os parâmetros de cada um de maneira específica. No caso da *Scattering*, temos que considerar a minimização do ITAE por meio dos seus parâmetros k, a, b, e c.

Já para o processo de controle Atraso deve-se utilizar a otimização considerando seus valores k, a e c apenas. Diferente do que foi utilizado nos outros modelos, para a *Smith Predictor* foi necessário gerar uma minimização com uma limitação no ganho já que se o ganho não tiver um valor limitado ele seria estimado em um valor extremamente grande, o que seria insustentável para o modelo. Depois de minimizar e encontrar os parâmetros, sendo

$$G_{c,\,sp}(s) = k\frac{s+a}{s+c} \tag{3.3}$$

Podemos descrever nosso controlador *Smith* como:

$$G_{sp}(s) = \frac{G_{c,\,sp}(s)}{1 + G_{c,\,sp}(s)\,G_p(s)\,[1 - e^{-sT}]} \tag{3.4}$$

3.2.3 Integração de Protocolos ZigBee e BACnet

Tendo como principal motivação o mercado de automação e as possibilidades de se desenvolver sistemas diretamente para esse mercado, foi estudado um modelo de integração dos protocolos para

que o sistema pudesse ter compatibilidade direta com o BACnet.

Para que essa integração pudesse ser realizada, primeiramente foi necessário analisar o processo de envio de mensagem do sistema em rede ZigBee com o objetivo principal de entender as limitações e as construções das mensagens utilizando uma determinada biblioteca.

Utilizando a API Bitcloud observou-se que o programa utiliza um padrão de comunicação específico onde os pacotes de dados são compostos de 8 bits. Para enviar mensagens de tamanho maior que os pacotes permitem, é necessário o uso de uma função de quebra de mensagem. Essa função permite que quando uma mensagem é recebida por um módulo e solicita envio, a função verifica por meio de alguns bits de controle o início e o fim da mensagem. Assim, a mensagem começa a ser quebrada em pacotes de 8 bits e enviada parte a parte para o módulo receptor. O módulo que está recebendo a mensagem espera até receber todos os pacotes que compõem a mensagem e fica com a responsabilidade de montá-la novamente[15, 32].

Com esse tipo de abordagem, conseguimos organizar a estrutura dos dados para que fossem enviados por meio desse método de envio de mensagens. Assim, com as mensagens estruturadas no modelo BACnet, conseguimos estabelecer o protocolo de comunicação desejado com o padrão BACnet.

A estrutura de mensagem do protocolo BACnet é composta de um array de 11 posições, como visto na Figura 3.23, onde as oito primeiras posições fazer parte do cabeçalho da mensagem e as três últimas fazem parte dos dados enviados na rede. A especificação BACnet define 8 tipos de frame de 00 a 07. Os tipos de 08 a 127 estão reservados para ampliações da especificação e os tipos 128 a 255 são reservados para frames específicos de cada fabricante[16]. Os tipos definidos para o frame estão definidos na Figura 3.24, onde é possível ver qual a função de cada um dos valores.

HEADER								DADOS		
0x55	0xFF	Tipo Frame	End destino	End fonte	Tamanho	Tamanho	CRC	dados	CRC	CRC

Figura 3.23: *Modelo do Frame do BACnet,[15].*

00 Token;
01 Poll for Master;
02 Reply to poll for Master;
03 Test Request;
04 Test Response;
05 BACnet data expecting Reply;
06 BACnet data not expecting Reply;
07 Reply Postponed;

Figura 3.24: *Funções dos Frames do protocolo BACnet,[15].*

Os frames tipo 00, 01 e 02 são interpretados somente pelo Device Coordenador, os mesmos não são interpretados pelos *End Device*[16].

O frame tipo Token (00): É utilizado na comunicação entre as estações mestres da rede, não apresenta dados, a estação mestre que está com o token pode iniciar a comunicação. O token é passado para outro Device mestre após o número máximo de dados for enviado pela rede [16].

O Frame tipo Poll for Master (01): Este frame é usado durante a configuração da rede, é utilizado para que sejam encontrados Devices novos na rede e determinar a sequência do Token, Neste Frame apenas as estações mestre respondem, as estações escravas devem ignorar o Frame.

O frame tipo Reply to Poll For Master (02): Este frame é utilizado para iniciar a comunicação na rede MS/TP, é utilizado para enviar uma informação particular a uma estação.

O frame tipo BACnet Data Expecting Reply (05): É usado por estações mestres para dados com parâmetros de um DL_UNITDATA.request, que contém endereço destino, endereço fonte, dados, prioridade e o código da mensagem, aguardando que a estação destino mande uma resposta de confirmação de recebimento da mensagem.

O frame tipo BACnet Data not Expecting Reply (06): Utilizado para transmitir dados de parâmetros de um DL_UNITDATA.request que contém endereço destino, dados, prioridade e código da mensagem, não aguarda resposta da estação destino.

O frame tipo Reply Postponed (07): É utilizado por estações mestre, com a função de indicar que a resposta a um frame Data Expecting Reply será enviada mais tarde na rede, não apresenta dados.

No protocolo BACnet os endereços destino e fonte são formados por dois bytes, destino e fonte, respectivamente. As posições tipo Tamanho (posição 6 e 7 do array da mensagem), da estrutura da mensagem, são formadas por dois bytes que informam a quantidade de bytes de dados da mensagem e dos dados a serrem enviados respectivamente[16].

O campo CRC cabeçalho da mensagem é a última parte presente no cabeçalho, é o campo para checagem de erros de transmissão. O método utilizado neste protocolo é o CRC-8 (*Cycling Redundancy Check*).

O campo Dados da mensagem pode apresentar 0 a 501 bytes, conforme especificação BACnet. No CFW-11 os dados podem apresentar até 59 bytes.

Os campos CRC da parte de dados da mensagem é a última parte do telegrama, sendo este o campo em que ocorre a checagem de erros de transmissão dos dados enviados na rede. O método utilizado é o CRC-16 (*Cycling Redundancy Check*)[16].

Na Figura 3.25 é mostrado uma comparação entre a arquitetura do protocolo BACnet e as camadas equivalentes no protocolo OSI.

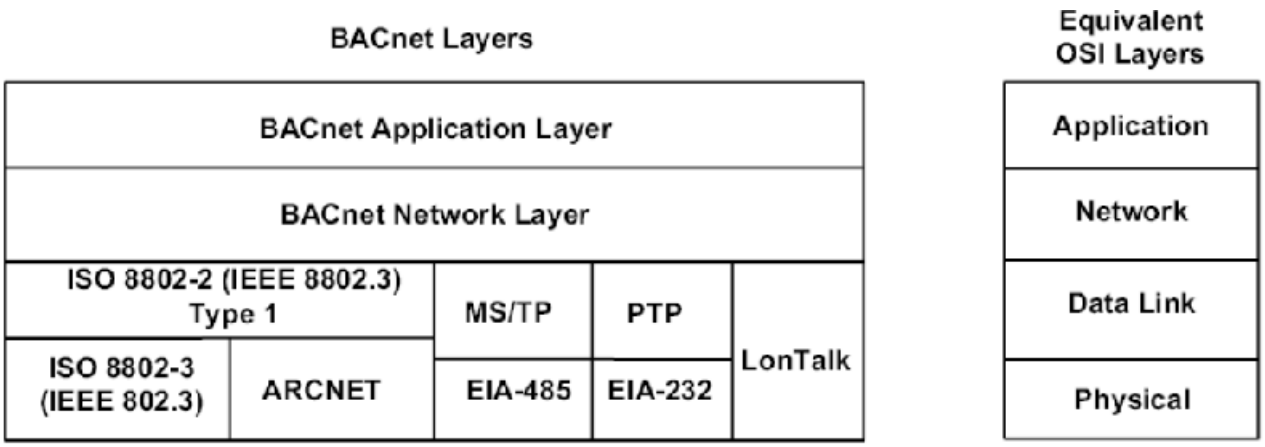

Figura 3.25: *Arquitetura do protocoloBACnet,*[16].

Afim de simplificar a implementação, neste projeto os únicos dados enviados pela rede, via BACnet, foram a temperatura e o dado de comando, comando este que é binário uma vez que o PWM, implementado em ambiente Simulink, controla o valor de comando das mensagens, ficando responsável por modular a tensão média aplicada ao atuador.

Portanto tem-se que a implantação de um protocolo de comunicação comercial torna a aplicação desenvolvida neste projeto mais próxima às soluções encontradas no mercado, fazendo com que a pesquisa desenvolvida neste projeto passa ser aplicada ao mercado de automação predial.

Abaixo é apresentada a Figura 3.26 que ilustra o dado de comando, valor zero ao fim do array na imagem, transmitido do Simulink para o nó coordenador.

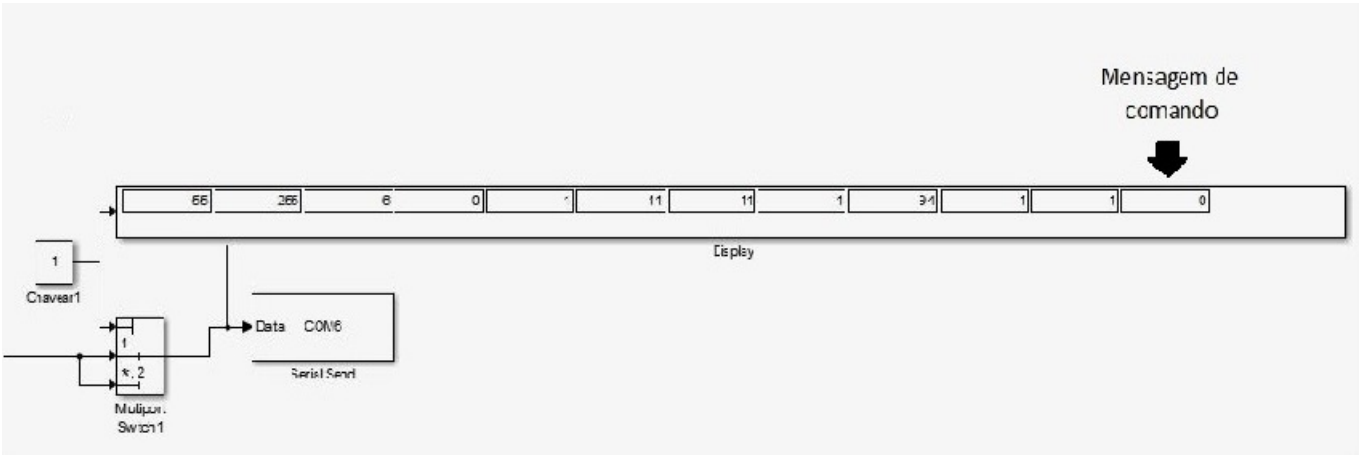

Figura 3.26: *Mensagem no gerada pelo Simulink com o dado de comando.*

A Figura 3.27 mostra, em ambiente Simulink, as posições que compõem a mensagem transmitida pela rede ZigBee transmitidas no formato BACnet durante um experimento de coleta de dados do trabalho.

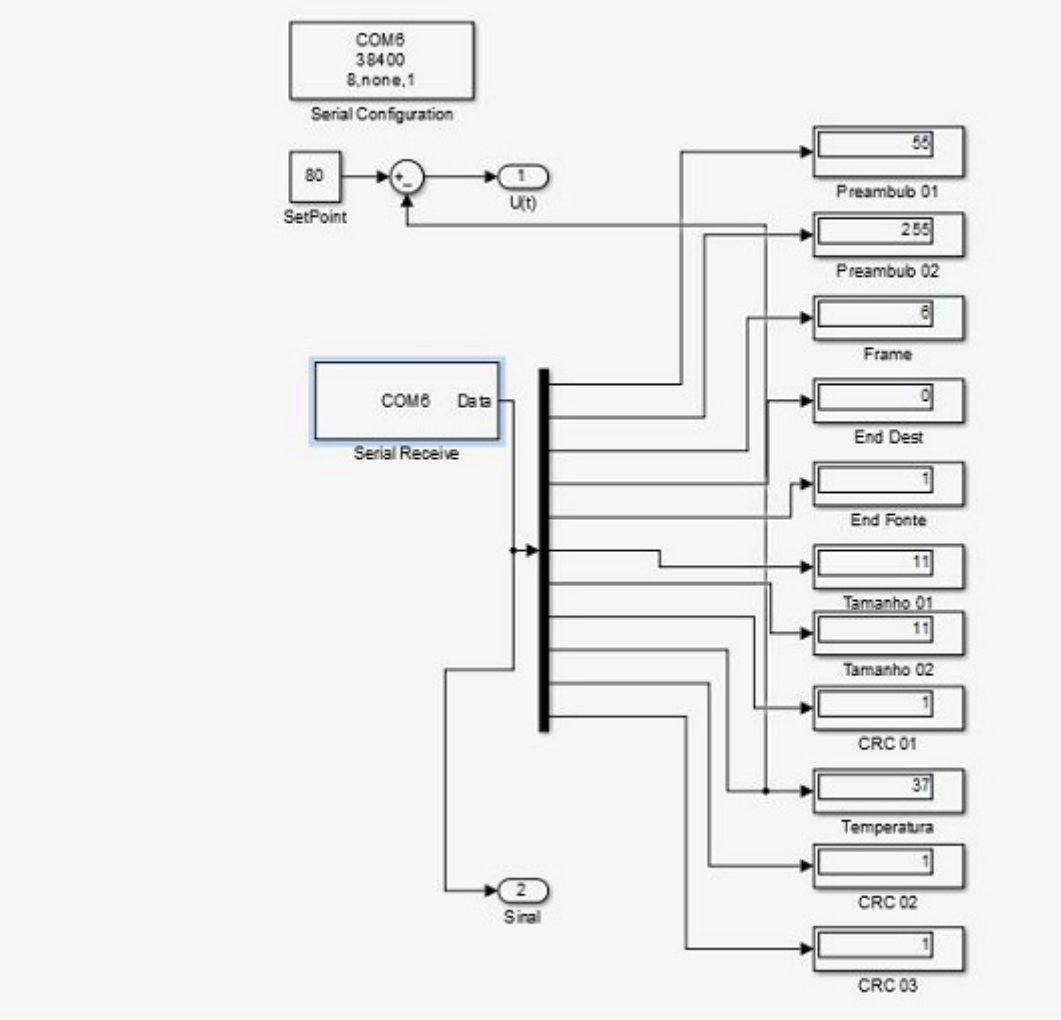

Figura 3.27: *Mensagem no formato BACnet enviada para o Simulink.*

É importante observar que o Simulink envia uma mensagem de 12 posições, sendo a última posição o dado referente ao comando do atuador, não configurando assim um protocolo de BACnet, porém o código implementado no nó coordenador remonta a mensagem de modo que ela siga novamente o formato padrão de mensagem do protocolo BACnet, sendo é assim é importante salientar que a mensagem é envida na rede wireless em padrão BACnet, porém a mensagem gerada pelo Simulink em função dos controladores não segue padrão BACnet, e portanto deve ser remontada no nó coordenador da rede deste trabalho.

Pode-se observar que as terminologias usadas em BACnet e ZigBee são diferentes com relação aos Devices da rede, o nó mestre referido em BACnet é o nó coordenador na topologia ZigBee e os nós tipo escarvo em BACnet são os nós tipo End Device em ZigBee.

Capítulo 4

Resultados

Serão apresentados os resultados gerais obtidos dos experimentos e simulações.

4.1 Rede de Automação

Considerando o projeto de sistema utilizando protocolo ZigBee, tinha-se como meta primária implementar uma rede consistente (estável), capaz de manter-se em comunicação com o sistema de processamento e com capacidade de mobilidade, flexibilidade, e que funcionasse durante o maior tempo possível.

Todos os experimentos realizados tiveram duração de 20 mim cada, como referência foi utilizado um Setpoint de $45°C$ para controle de temperatura, onde foi coletado dados gráficos da temperatura, do erro da temperatura em relação à referência, e das simulações dos três modelos de controladores implementados afim de provar o melhor desempenho do controle por *Scattering* frente aos demais controladores.

4.2 Modelagem experimental

Considerando o modelo da maquete de automação predial e verificando experimentalmente os modelos de controle implementados em Simulink pôde-se coletar e analisar os resultados obtidos dos experimentos. Foram consideradas duas posições para o sensor diferentes para provocar uma diferença no atraso de tempo e causar uma pequena pertubação no sistema.

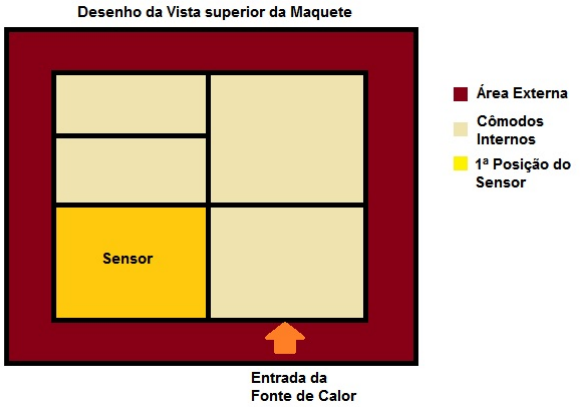

Figura 4.1: *Vista Superior Esquemática da Maquete - Primeira posição.*

Para abordagem experimental, consideramos três conjunto de valores das constantes experimentais utilizadas, são elas:

$$b \rightarrow Constante\ referente\ à\ Transformada\ de\ Scattering$$

$$T_d \rightarrow Atraso\ de\ tempo\ variável$$

$$T_{dn} \rightarrow Atraso\ nominal\ do\ sistema.$$

Assim, organizamos a comparação dos três modelos de controle para cada uma das combinações de constantes, como mostrado a seguir:

Parâmetros	b	Td	Tdn
Combinação 1	1,5	0,1	0,1
Combinação 2	1,95	0,1	0,1
Combinação 3	1,95	0,2	0,4
Combinação 4	2,5	0,1	0,1

Tabela 4.1: *Combinações de Constantes da Transformação de Scattering.*

De forma a organizar o conjunto de dados é apresentado uma comparação gráfica para análise comparativa entre a resposta térmica do sistema e a comparação entre os controladores para as combinações de parâmetros escolhidos para validar este trabalho.

- Análise para a combinação 2

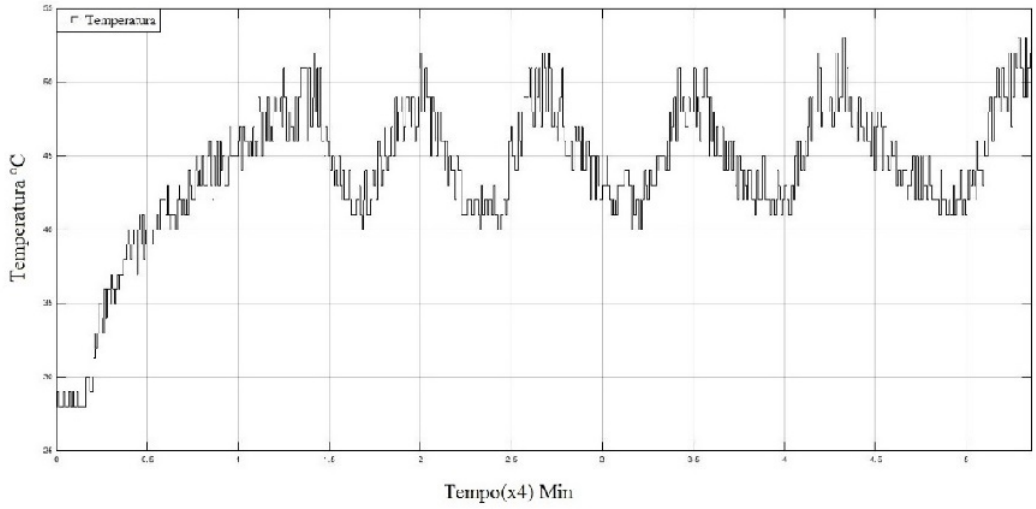

Figura 4.2: *Combinação 2 - Controlador Atraso, Temperatura (ºC) vs. Tempo (x4) min.*

Na Figura 4.2 observa-se que a temperatura varia em de 4°C em torno do Setpoint experimental, quando o controlador do tipo Atraso é utilizado na malha de controle.

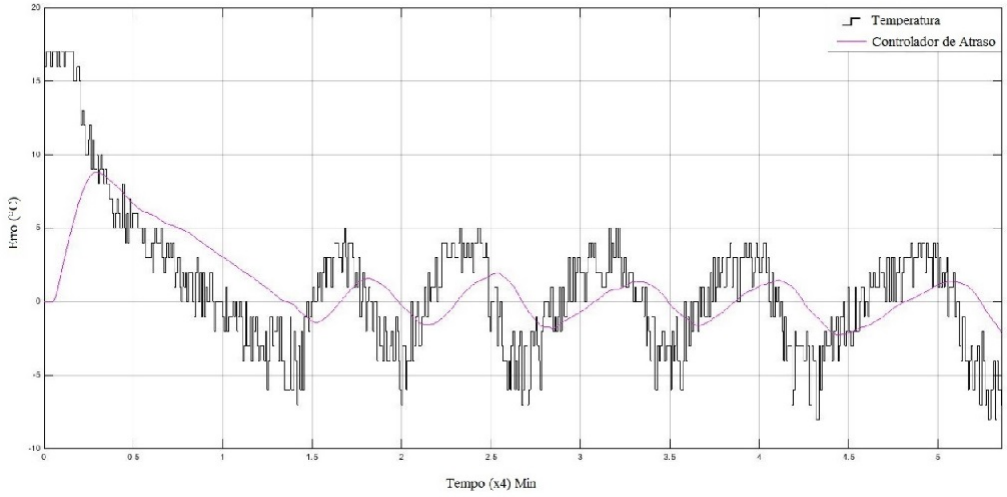

Figura 4.3: *Combinação 2 - Controlador Atraso Ajustado, Erro (ºC) vs. Tempo (x4) min.*

Na Figura 4.3 pode-se observar o erro entre a temperatura medida pelo sensor e o *Setpoint* desejado para o controlador de Atraso.

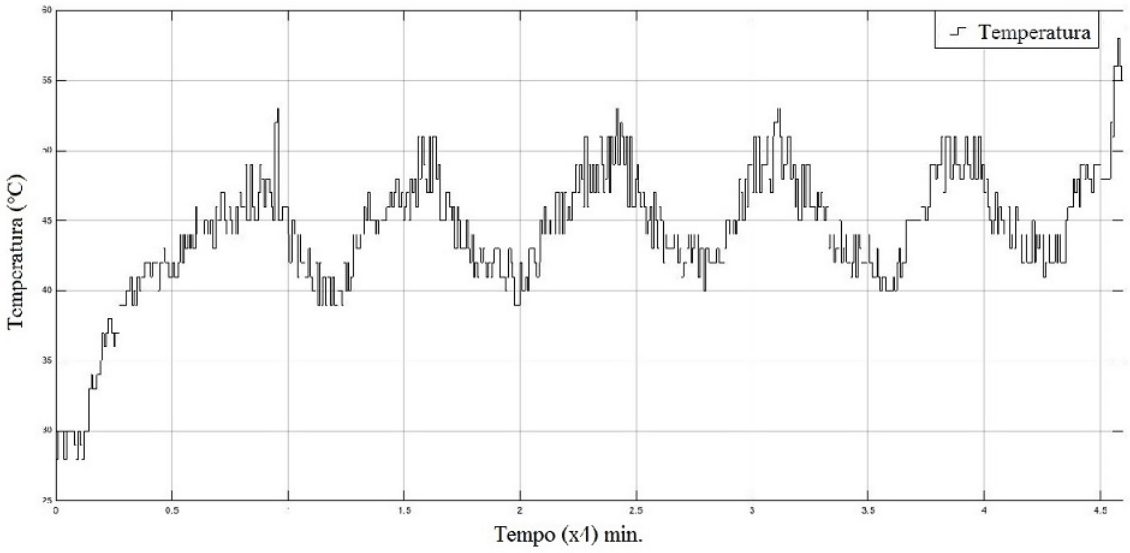

Figura 4.4: *Combinação 2 - Controlador Scattering, Temperatura (ºC) vs. Tempo (x4) min.*

Na Figura 4.4 observa-se que com a implementação do controlador Scattering a temperatura ficou mais próxima ao Setpoint, a variação se deu entorno de 3°C.

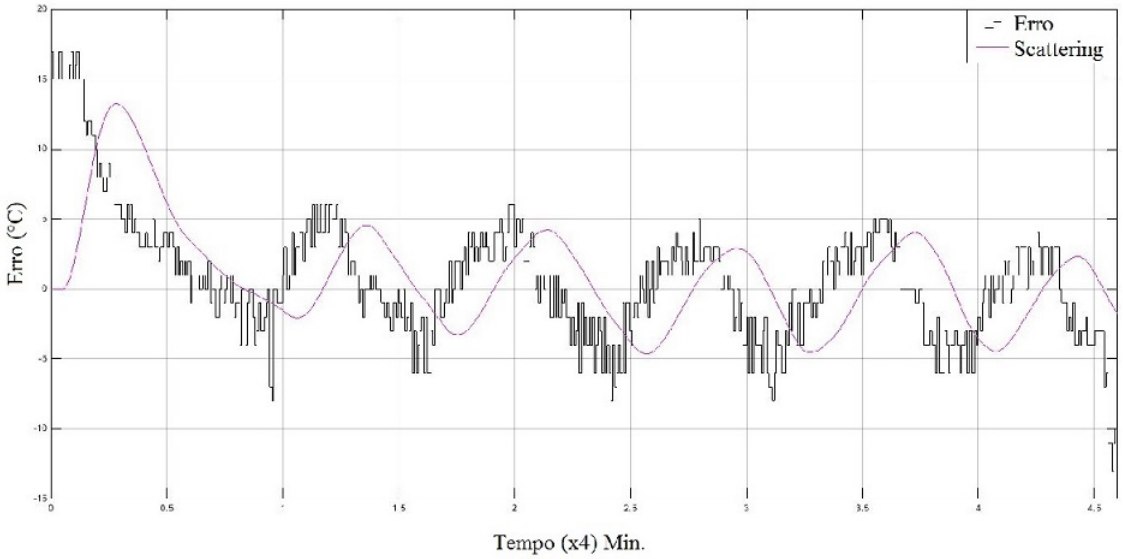

Figura 4.5: *Combinação 2 - Controlador Scattering, Erro (ºC) vs. Tempo (x4) min.*

A Figura 4.5 mostra a resposta do controlador Scattering em função do erro entre a temperatura e o *Setpoint* durante o experimento.

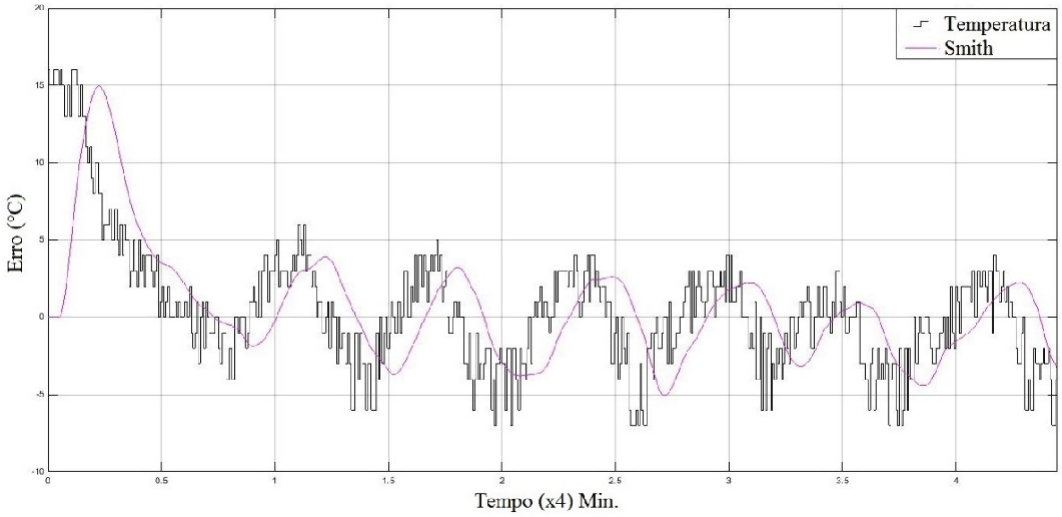

Figura 4.6: *Combinação 2 - Controlador Smith Predictor, Erro (ºC) vs. Tempo (x4) min.*

Na Figura 4.6 temos o erro em °C da diferença ente o *Setpoint* e o sinal de controle do *Smith Predictor*, onde podemos observar que o erro médio se dá em torno de 4 graus.

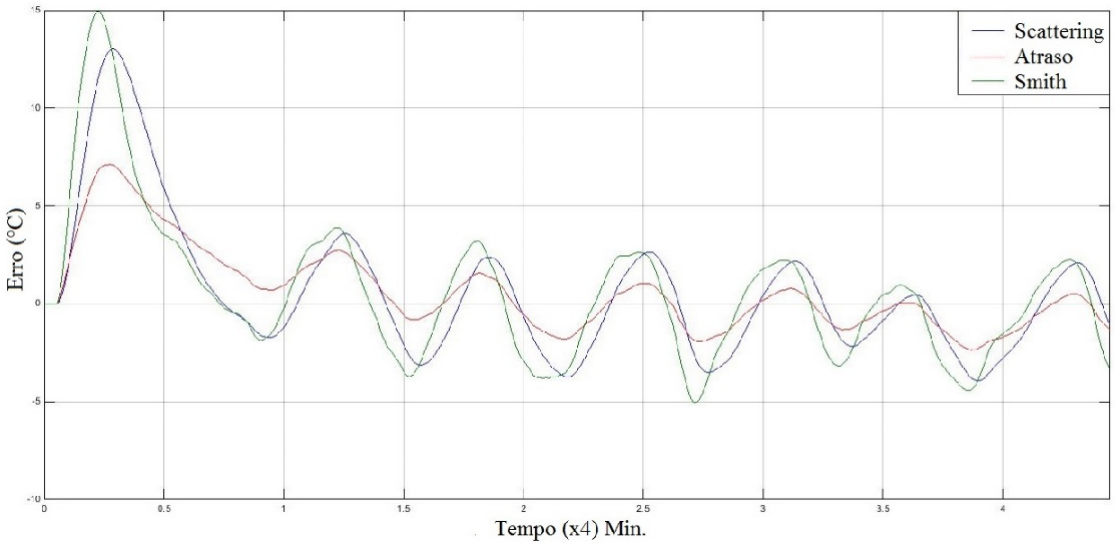

Figura 4.7: *Combinação 2 - Três Controles Sobrepostos, Erro (ºC) vs. Tempo (x4) min.*

Na Figura 4.7 pode-se observar a resposta dos três controladores quando o controlador *Smith Predictor* atuou no controle da temperatura.

A partir da análise dados, foi observado que os controladores mais dependentes do atraso de tempo foram o Atraso e o Smith, apesar de terem obtido resultados mais rápidos em comparação com a Scattering. Outro ponto de interesse foi que o controle de Atraso tem uma resposta mais rápida em um primeiro momento e com menor sobressinal associado, porém tem resposta mais lenta ao longo do tempo de execução da simulação.

Ao longo do experimento foram realizadas perturbações manuais na maquete de automação onde foram abertas portas e/ou janelas para alterar os valores do atraso da rede, sendo assim observa-se que apesar de nenhum dos controladores terem perdido qualidade significativa no controle da temperatura, a scattering foi o controlador que implementou um controle mais suave e preciso em torno da temperatura desejada frente aos demais controladores.

A seguir será analisado os dados obtidos com a combinação 3, onde os atrasos T_d *e* T_{dn} tiveram seus valores alterados de 0.1 para 0.2 *e* 0.4 respectivamente e o parâmetro "b" da scattering manteve-se 1.95.

- Análise para a combinação 3

Na figura 4.8 está ilustrado a variação da temperatura em função do controle realizado pelo controlador tipo Atraso, pode-se observar que há uma piora no controle, pois a temperatura oscila um valor de $5°C$ em torno da referência ($45°C$).

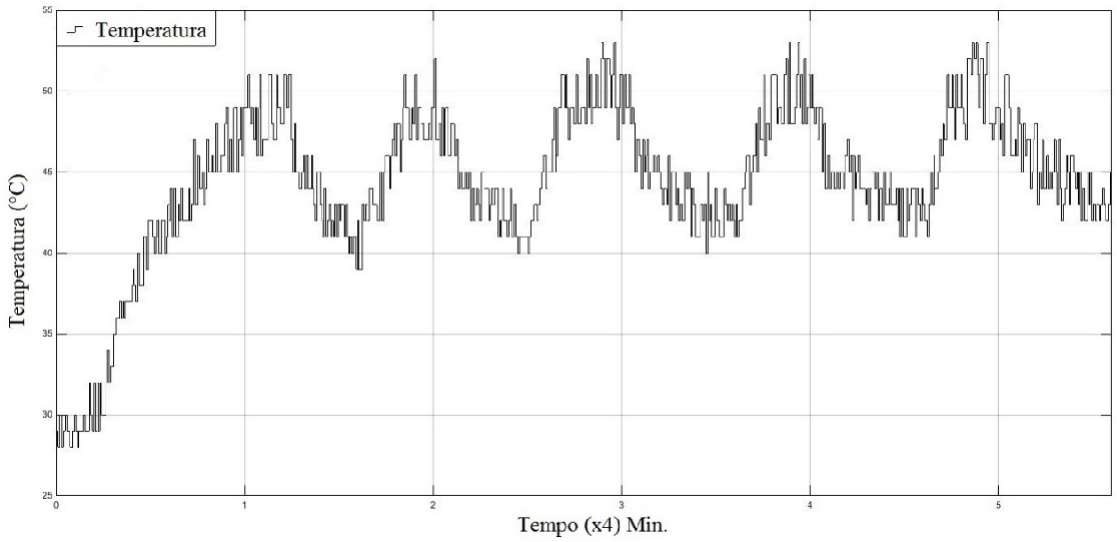

Figura 4.8: *Combinação 3 - Controlador tipo Atraso ajustado, Temperatura (ºC) vs. Tempo (x4) min.*

No gráfico apresentado na Figura 4.9 está mostrado o erro da temperatura, diferença entre a temperatura do ambiente e o Setpoint desejado, e os valores de controle do controlador atraso durante o experimento.

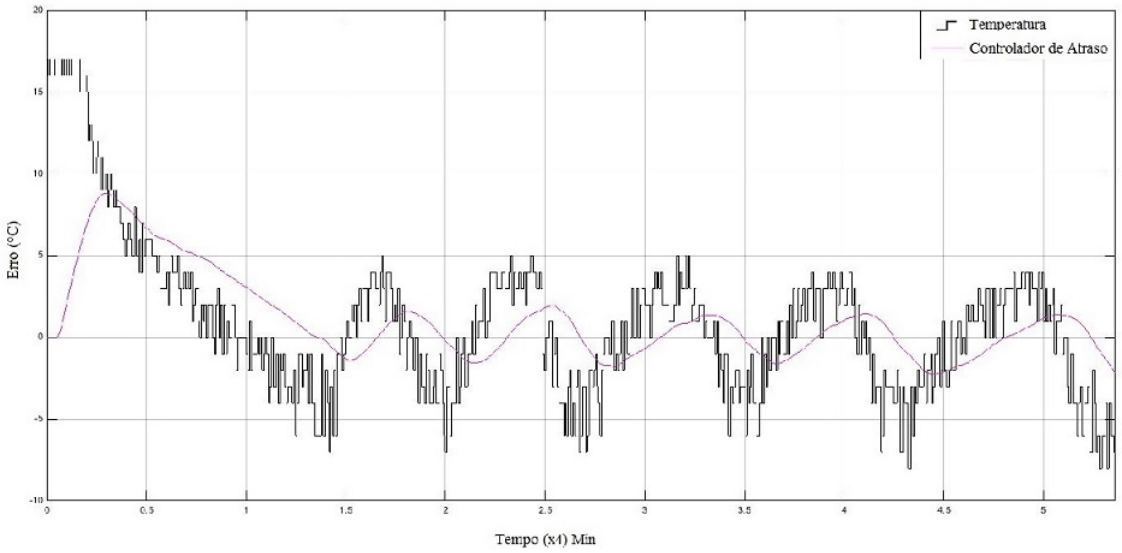

Figura 4.9: *Combinação 3 - Controlador Atraso ajustado, Erro (ºC) vs. Tempo (x4) min.*

No gráfico apresentado abaixo, observa-se que a Scattering, apesar de ter seu desempenho afetado pelos atrasos, realiza o controle da temperatura com um erro inferior ao dos controladores tipo Atraso e *Smith Predictor*, observa-se que ao final do experimento, quando o sistema tende a se tornar estacionário, o erro se dá em torno de $3°C$ em torno do *Setpoint* desejado.

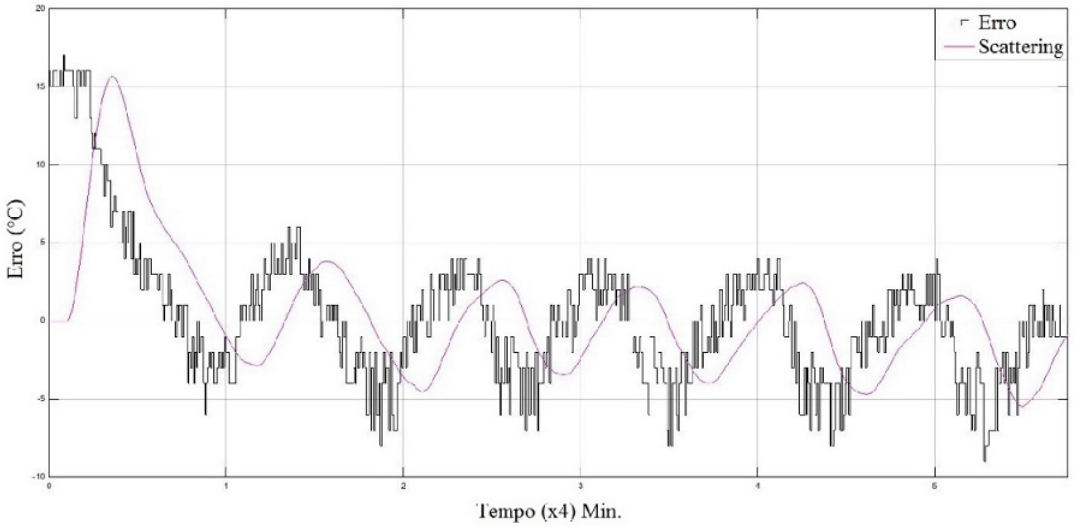

Figura 4.10: *Combinação 3 - Controlador Scattering, Erro (ºC) vs. Tempo (x4) min.*

Para esta combinação dos parâmetros T_d, T_{dn} *e* "b", o gráfico do erro entre a temperatura e o *Setpoint* mostrado na figura a seguir, prova que o controlador tipo *Smith Predictor* não consegue realizar um controle de temperatura caso os atrasos sejam variados.

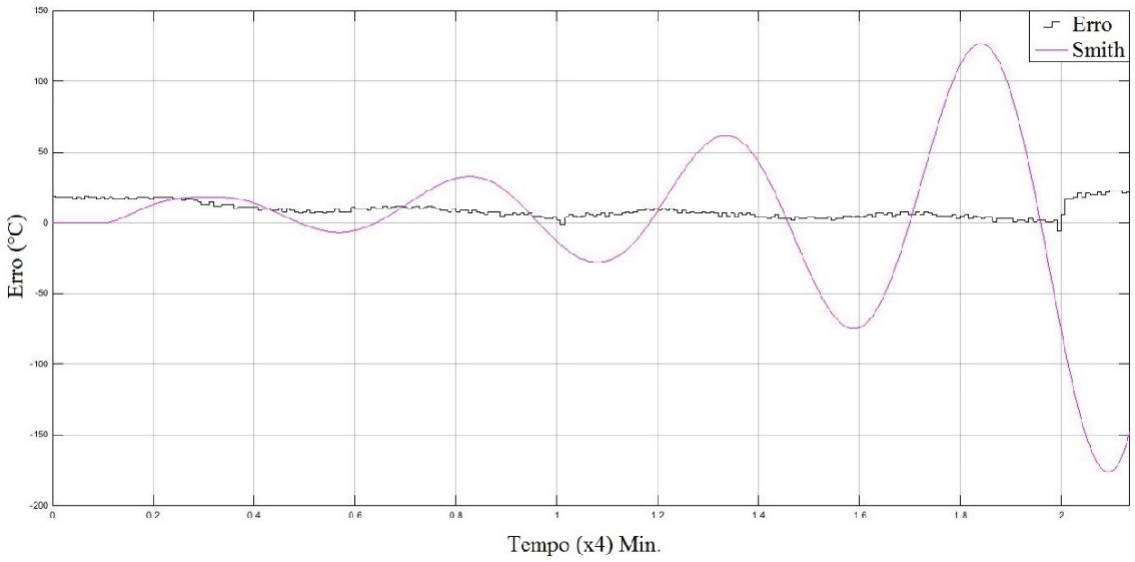

Figura 4.11: *Combinação 3 - Controlador Smith Predictor. Erro (°C) vs. Tempo (x4) min.*

Analisando os gráficos apresentados, pode-se observar que para cada um dos sinais a modificação no tempo de atraso teve uma consequência diferente. Em primeiro lugar, observa-se que o controlador tipo atraso obteve uma pequena melhora no tempo de subida, e em contrapartida apresentou um maior sobressinal.

Já a transformada de *Scattering* apresentou a mudança mais sutil dentre os três modelos. Observa-se que, como comprovado teoricamente e por meio de simulações a transformada de *Scattering* apresenta uma menor sensibilidade à mudança do tempo de atraso.

Por último, a *Smith Predictor* apresentou um péssimo resultado, confirmando sua alta dependência de modelagem em cima de um atraso de tempo constante e conhecido, mostrando que dentre os três controladores o único que responde bem à mudança de atrasos é a *scattering*, pois os demais controladores ou pioram bastante o controle realizado ou não consegue realizar o controle do sistema.

Faz-se então, por meio do gráfico a seguir, uma comparação entre as três respostas apresentadas pelos controladores para estes parâmetros de teste.

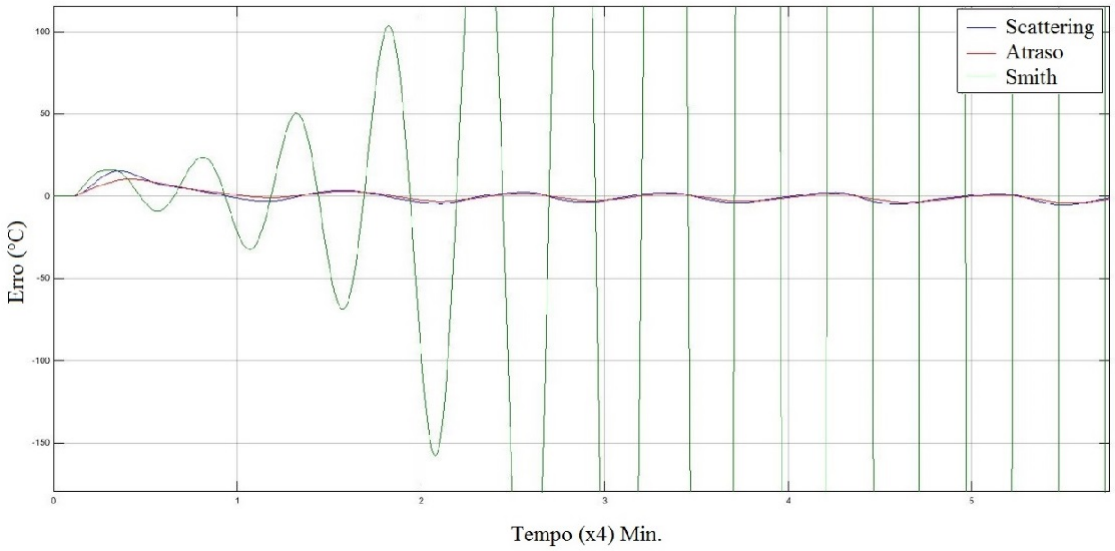

Figura 4.12: *Combinação 3 - Conjunto de Controladores. Erro (ºC) vs. Tempo (x4) min.*

Obserserva-se que o desempenho da Scattering foi pouco alterado com relação ao desempenho dos outros dois modelos no que se refere ao controle da temperatura, o que comprova que os modelos mais dependentes do tempo de atraso possuem uma certa dificuldade quando se trata de manter seu desempenho e resultado para sistemas de controle em rede (NCS).

Na Figura 4.13 foi dado um zoom, na parte inicial do gráfico anterior, para ilustrar melhor o comportamento dos controladores quando o controlador Smith Predictor não conseguiu mais realizar o controle.

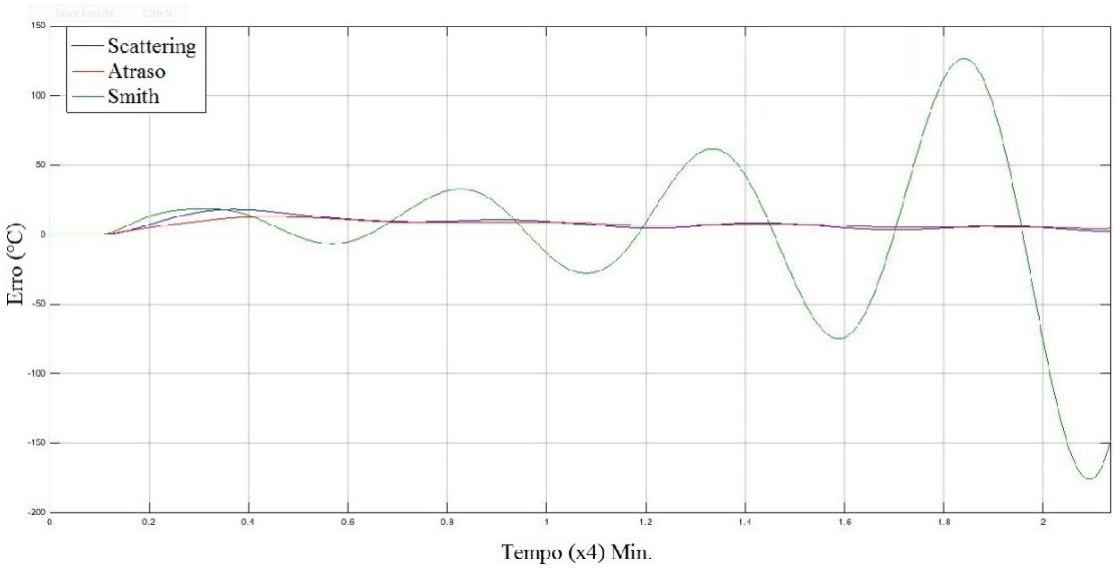

Figura 4.13: *Combinação 3 - Conjunto de controladores, Erro (ºC) vs. Tempo (x4) min.*

Deduz-se pelos gráficos que as mudanças no atraso influenciaram bastante no comportamento dos controladores tipo Atraso e *Smith Predictor*, e que apesar de ter seu comportamento um pouco

piorado devido à mudança nos atrasos de projeto, o controlador *scattering* conseguiu realizar um controle aceitável de temperatura em torno do *Setpoint*, o que prova que para um ambiente de automação predial o uso de controlador *scattering* se mostra mais eficiente no controle de temperatura, mas não somente para temperatura como também para qualquer sistema de controle em que os atrasos sejam variáveis, o uso de *Scattering* é extremamente melhor, sendo ideal para um controle de processos críticos, os quais apresentem variações de atrasos e necessitem de um controle robusto e preciso, que não tenha que ser adaptado para cada situação ou interferência que ocorrer no sistema, como é por exemplo o controle de uma caldeira de água em uma usina nuclear.

A seguir serão apresentados os dados referentes à combinação um, onde será mostrado que o coeficiente "*b*" da *Scattering* com valor de 1.5 torna a mesma um pouco pior se comparada ao controle quando aquele coeficiente tem valor 1.95, as combinações um, dois e quatros foram realizadas em experimento afim de provar que o valor ótimo para o parâmetro "*b*" da *Scattering* é 1.95.

- Análise para a combinação 1

No gráfico abaixo, é mostrado como a temperatura do sistema na maquete de automação predial varia em função do controle realizado pelo controlador *Scattering*.

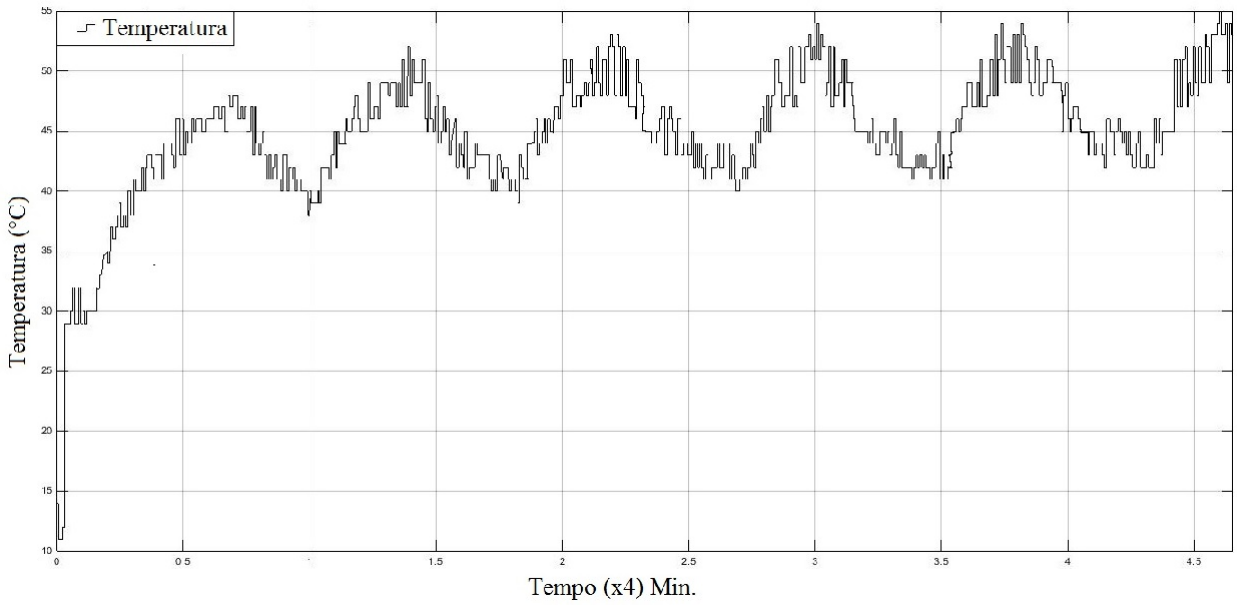

Figura 4.14: *Combinação 1 - Controlador Scattering, Temperatura (ºC) vs. Tempo (x4) min.*

Considerando a combinação número um observa-se uma modificação nos parâmetros específicos da transformada de *Scattering*, isto é, modificamos a constante característica desse modelo para observar possíveis variações de resposta.

No Gráfico a seguir, tem-se uma comparação entre o comportamento dos três controladores, onde pode-se observar, através da comparação entre este gráfico e os das outras combinações anteriores, que o controlador *Scattering* responde pior com este parâmetro, não observa-se mudança

no comportamento dos outros controladores devido ao parâmetro "*b*" em questão afetar somente o comportamento da *Scattering*.

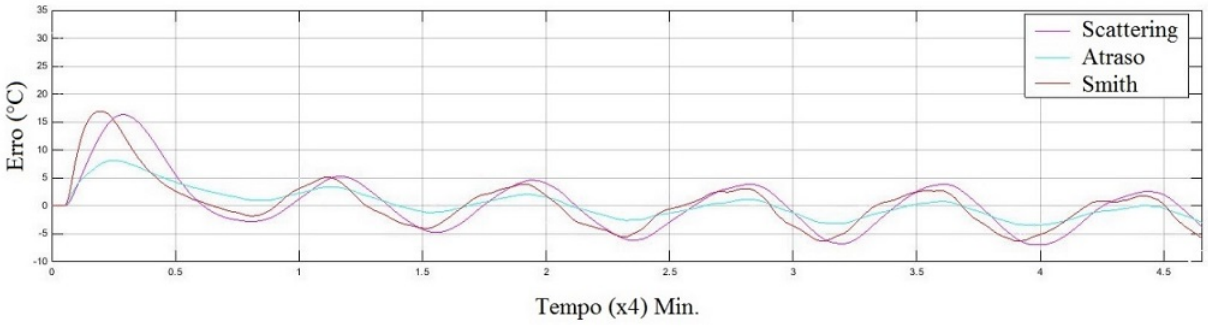

Figura 4.15: *Combinação 1 – conjunto de controladores, Erro (°C) vs Tempo (x4) Min.*

Observado agora os três resultados para a combinação número 1, tem-se claramente que a transformada de *Scattering* foi afetada pela modificação de seu parâmetro "*b*". Assim, comprova-se experimentalmente que, como na teoria, a transformada de *Scattering* mantém-se pouco afetada com relação ao tempo de Atraso e suas variações, no entanto caracteriza uma relação de dependência com o parâmetro "*b*" analisado.

A seguir será apresentada a última combinação de parâmetro utilizadas nesse trabalho, onde o parâmetro "*b*" da *Scattering* foi modificado para o valor 2.5 e será analisado o comportamento da mesma em função da variação deste valor.

- Análise para a combinação 4

Observa-se através da Figura 4.16 que o controle de temperatura (realizado pela *Scattering*) é pior se comparado ao mesmo controle realizado quando seu parâmetro tem valor 1.95.

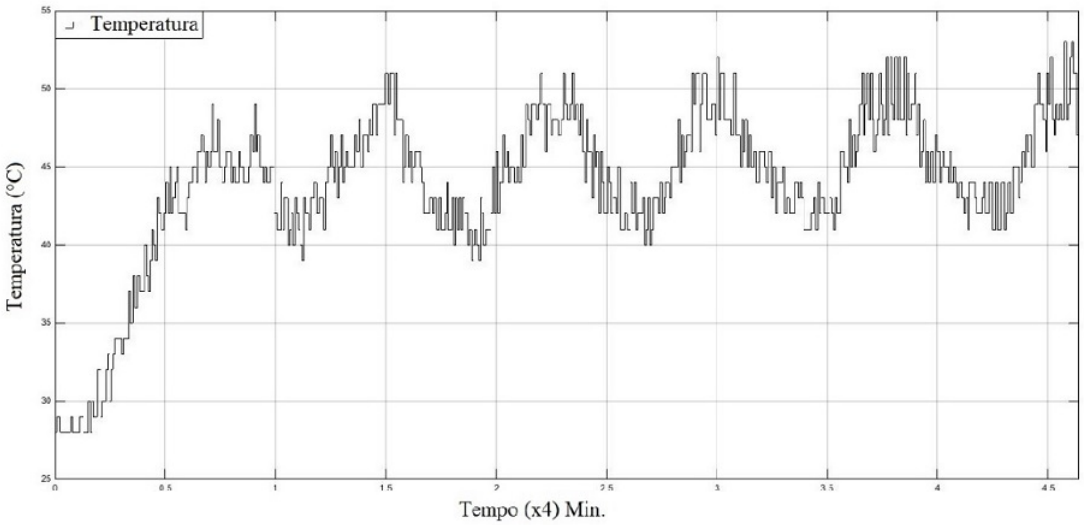

Figura 4.16: *Combinação 4 - Controlador Scattering, Temperatura (°C) vs. Tempo (x4) Min.*

No próximo gráfico, pode-se observar a variação do Erro em $^{\circ}C$e o sinal de controle gerado pela *Scattering*, pode-se observar que o erro se dá de $5^{\circ}C$ em torno do *Setpoint* desejado.

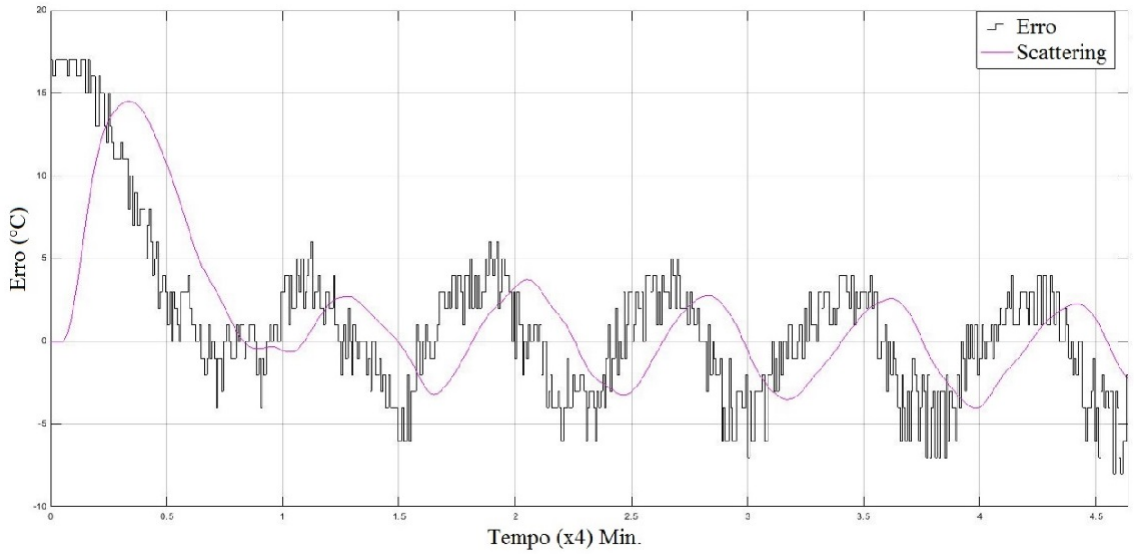

Figura 4.17: *Combinação 4 - Controlador Scattering. Erro (ºC) vs. Tempo (x4) min..*

Portanto observou-se que essa mudança do parâmetro "*b*" da *Scattering* provoca um menor tempo de subida, e consequentemente, um sobressinal reduzido, porém um erro observado um pouco maior que o apresentado quando utilizado os parâmetros anteriores de *"b"*.

No gráfico da figura 4.17 será mostrado o comportamento dos três controladores a título de comparação do modelo *Scattering* com relação aos outros dois, em função da mudança do valor do parâmetro "*b*" para 2.5.

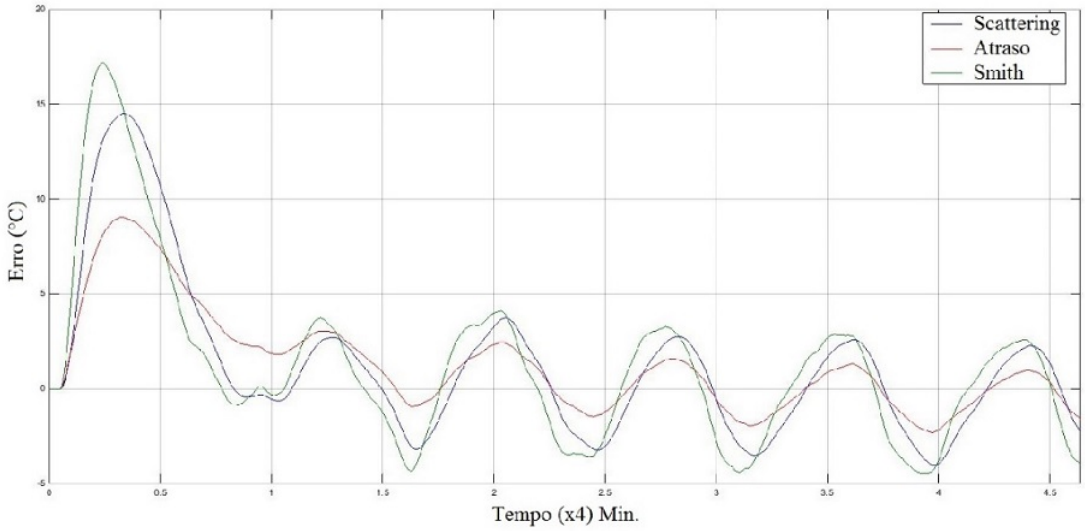

Figura 4.18: *Combinação 4 - Conjunto de controladores. Erro (ºC) vs. Tempo (x4) min.*

Comparativamente uma mudança nos parâmetros da *Scattering* possibilitou sua piora de de-

sempenho, pois o erro em função da diferença entre a temperatura na maquete de automação predial e o *Setpoint* desejado aumenta, embora ainda assim o modelo tenha superado os resultados observados pelo sistema *Smith*. Temos um menor sobressinal advindo do controle por Atraso em comparação com os demais, e um menor tempo de pico característico da *Smith Predictor*. Assim, vemos que a transformada de *Scattering* se mantém com um sobressinal e um tempo de pico intermediário ente os dos outros dois modelos, porém apresenta um controle mais eficiente no que se refere às variações do ambiente que causa atrasos na transmissão térmica do atuador.

Pode-se inferir então que ao tratar-se de modelos mais dependentes do atraso no tempo (*Smith Predictor* e Atraso), tem-se que considerando as variações e inconstâncias nessas variáveis de atraso de tempo, que a modelagem utilizando métodos menos influenciados pelo atraso no tempo, permite que o controle do sistema mais eficiente no que desrespeita a manter o controle em torno de uma referência, que neste caso é o *Setpoint* de temperatura.

Comparativamente uma mudança nos parâmetros da *Scattering* possibilitou sua piora de desempenho, embora ainda assim o modelo tenha superado os resultados observados pelo sistema Smith. Temos um menor sobressinal advindo do cotrole por PID em comparação com os demais, e um menor tempo de pico característico da Smith *Predictor*. Assim, vemos que a transformada de *Scattering* tenta se manter com um sobressinal e um tempo de pico médio.

Podemos então consluir que ao tratarmos de modelos dependentes do atraso no tempo, temos que considerar a cada momento as variações e inconstâncias nessas variáveis, que são tangíveis de modificação por um agente externo ou por uma modificação de meio e arquitetura. Enquanto isso, a modelagem utilizando métodos não dependentes de atraso no tempo, permite que a sensibilidade do sistema seja focada em uma diferente variável de configuração.

Capítulo 5

Conclusões

De acordo com a proposta do projeto, dever-se-ia implementar um NCS, tal que fossem utilizadas três modelagens de controle para avaliar o sistema e suas características.

Além disso, o projeto envolveu o planejamento de uma rede de automação em que se pudesse adaptar as necessidades do mercado, e atender aos pré-requisitos necessários para o funcionamento do NCS. Dentre as possibilidades encontradas, foi desenvolvido o sistema a partir do protocolo de rede ZigBee.

A partir da identificação da planta, foi possível realizar um modelamento do sistema baseado no método de otimização por critério de ITAE (Integral Time-weighted Absolute Error)[31]. Utilizando as três modelagens distintas, duas delas caracterizam modelos de controle extremamente dependentes do atraso no tempo, senso elas o controlador tipo Atraso e o *Smith Predictor*. A terceira, e principal alvo do estudo proposto é a transformada de *Scattering* que apresenta uma solução menos dependente da variação do atraso no tempo.

Uma das características limitantes do projeto foi que o sistema se comunica por meio de pacotes de 8 bits, o que limitou de certa maneira a obtenção de uma resolução co mamior definição nos gráficos, mas que se considerando um tempo de resposta longo graças ao sistema de propagação de calor ser lento, não provocou problemas na interpretação dos gráficos.

Um dos objetivos visava a concreta integração do sistema ZigBee com o protocolo de comunicação mais comum no mercado hoje sendo ele o BACnet. Nesse processo, modificou-se a estrutura da mensagem enviada pela rede para que fosse possível enviar ,por meio de pacotes de dados, mensagens que originalmente não pertenciam a este protocolo de comunicação mas sim ao protocolo de comunicação original do ZigBee. Assim, tornou-se possível a implementação de uma integração BACnet, tal que as mensagens enviadas pelos módulos na rede fossem no formato do protocolo BACnet.

5.1 Perspectivas Futuras

Considerando o projeto desenvolvido, observou-se que para cada novo objetivo alcançado, durante a implementação do projeto, novas portas de pesquisa e estudo eram abertas. Assim, o projeto deixa como legado, não só a análise e os resultados das simulações, mas também algumas propostas de desenvolvimento e pesquisas futuras que podem ser utilizadas.

- Em primeiro lugar cita-se a mudança de resolução de leitura do sensor analógico para 10 bits, ou mudança para uso de um sensor digital, como sugestão SHT71, permitindo assim uma leitura mais precisa, menos custosa, e claramente com mais detalhes, permitindo uma análise melhorada;
- Um outro quesito a ser modificado é a possibilidade de testes reais dos três modelos abordados (Transformada de *Scattering*, *Smith Predictor* e PID especificamente ajustado) numa sala ou no próprio laboratório do LARA. Possivelmente a demonstração do impacto do projeto numa situação tipicamente real do dia a dia pode trazer resultados mais interessantes do ponto de vista de aplicação real e possibilidade de um produto no mercado com essa tecnologia;
- Uma possível implementação de interesse é o uso de equipamentos tipicamente especificados para um certo protocolo de automação do mercado, como por exemplo MODBUS ou BACnet e a tentativa de integração entre eles e o sistema em rede utilizando ZigBee, objetivando sempre a proposta de se adaptar ao mercado de forma a ter produtos eficientes, com custo reduzido e boa qualidade, promovendo-se como um produto competitivo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BAS and Control Systrems. http://www.kmccontrols.com/products/Understanding_Building_Automation_and_Control_Systems.aspx. Accessed: 2015-07-01.

[2] CIRURGIA Robótica. http://mauriciorubinstein.com.br/sobre-o-robo. Accessed: 2015-07-01.

[3] ZIGBEE Wireless Networking Solution. http://www.em.avnetasia.com/products/PromotionDetail.aspx?pagesId=26&MasterPageId=17&id=77. Accessed: 2015-07-01.

[4] XBEE Applications. http://www.digi.com/lp/xbee. Accessed: 2015-07-01.

[5] ZIGBEE Technologie. http://www.takeoffprojects.com/embd-con-zigbee-voice.php. Accessed: 2015-07-01.

[6] ARAUJO, H. A. S. de; MELO, M. C. C. *Controle adaptativo da climatização predial para eficiência energética*. 2013. Universidade de Brasília.

[7] MATIAKIS, T.; HIRCHE, S.; BUSS, M. Control of networked systems using the scattering transformation. *IEEE Conference on Control Applications*, 2005.

[8] INTEGRAçãO Multiprotocolo. http://www.oemctrl.com. Accessed: 2015-07-01.

[9] TOPOLOGIES. http://www.jennic.com/elearning/zigbee/files/html/module2/module2-3.htm. Accessed: 2015-07-01.

[10] ATMEL. *Zigbit 2.4Ghz Wireless Modules - ATZB-24-A2/B0 - DATASHEET*. [S.l.], 2013.

[11] SEMICONDUCTOR, N. *LM35 Precision Centigrade Temperature Sensors*. [S.l.], 2000.

[12] ATMEL. *AVR2050: BitCloud Developer Guide*. [S.l.], 2015.

[13] ZIGBEE. http://www.ZigBee.org. Accessed: 2015-07-01.

[14] RELE 8 Canais. http://profclaudiosilva.com.br/?p=325. Accessed: 2015-07-01.

[15] WEG. *Manual do Usuário BACnet CFW-11*. [S.l.], 2014.

[16] WEG. *Manual do Usuário CFW701*. [S.l.], 2012.

[17] BAS Market Grow. http://www.pacetoday.com.au/news/building-automation-and-controls-market-to-reach-$. Accessed: 2015-07-01.

[18] ACHIEVING Energy Savings with BAS. http://www.automatedbuildings.com/news/apr07/articles/esource/070322105430kamm.htm. Accessed: 2015-07-01.

[19] SOFTWARE United Technologies. http://www.automatedlogic.com/software. Accessed: 2015-07-01.

[20] THE Benefits of Wireless Controls for Building Automation. http://facilitymanagement.com/articles/buildingdesign-2014-8-01.html. Accessed: 2015-07-01.

[21] SISTEMA HVAC. http://www.tudoemfoco.com.br/hvac-o-que-e.html. Accessed: 2015-07-01.

[22] CONTROL of Thermal Power Plants. http://www.fepc.or.jp/english/environment/asia-pacific/green_handbook_peer/__icsFiles/afieldfile/2008/10/20/chapter2_1.pdf. Accessed: 2015-07-01.

[23] ARTL, A. B. et al. Passivity framework and traffic reduction for the teleoperation of a gantry crane. 2013.

[24] MATIAKIS, T.; HIRCHE, S.; BUSS, M. Independent-of-delay stability of nonlinear networked control systems by scattering transformation. *American Control Conference*, 2006.

[25] MILLAN, Y. A. et al. A wireless networked control systems review. 2011.

[26] BUILDING automation migrates towards Ethernet and wireless. http://www.iebmedia.com/wireless.php?id=8593&parentid=74&themeid=255&hft=69&showdetail=true&bb=1. Accessed: 2015-07-01.

[27] MATOS, L. G. de; MELO, N. C. de. *Economia de energia e gravação remota de dispositivos ZigBee visando a automação predial.* 2013. Universidade de Brasília.

[28] TANENBAUM, A. S. *Redes de Computadores.* [S.l.]: Editora Campus, 2003.

[29] VERMA, P. *802.15.4.* 2005. Samer Shammaa Telecomunication Engeneer Department.

[30] ATMEL. *Atmel AVR2052: Atmel BitCloud Quick Start Guide.* [S.l.], 2011.

[31] POSTLETHWAITE, D. B. *Measures of controlled system performance.* 2013. University of Strathclyde.

[32] MARTOCCI, J. P. Bacnet unplugged zigbee® and bacnet connect. *ASHRAE Journal*, 2008.

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

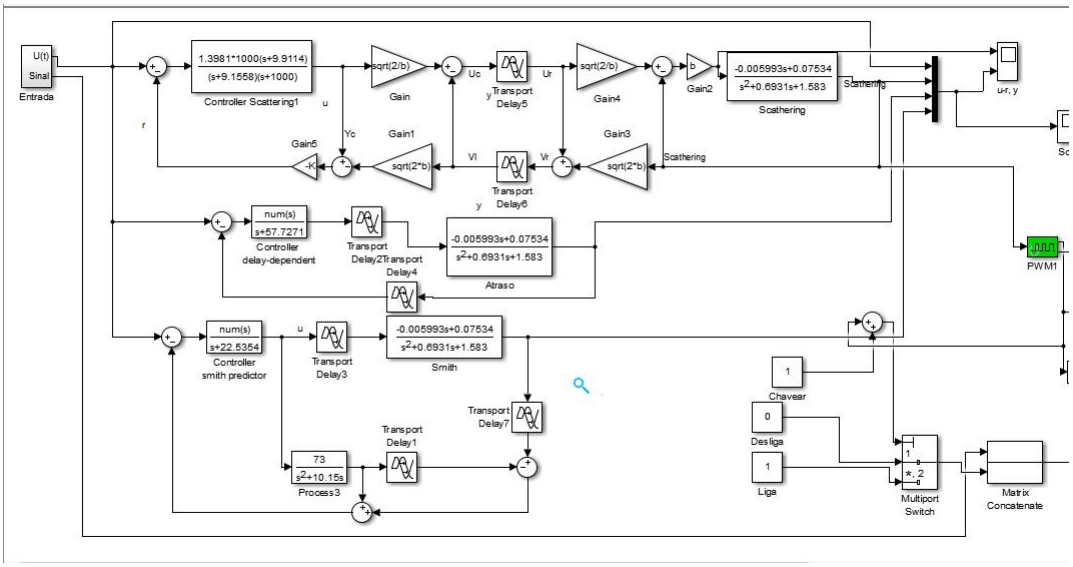

Figura I.1: Modelo em Simulink utilizando PWM na saída Scattering.

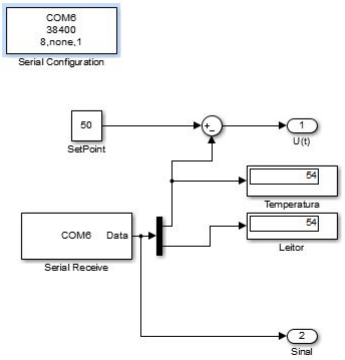

Figura I.2: Entrada de Referência subtraida do retorno.

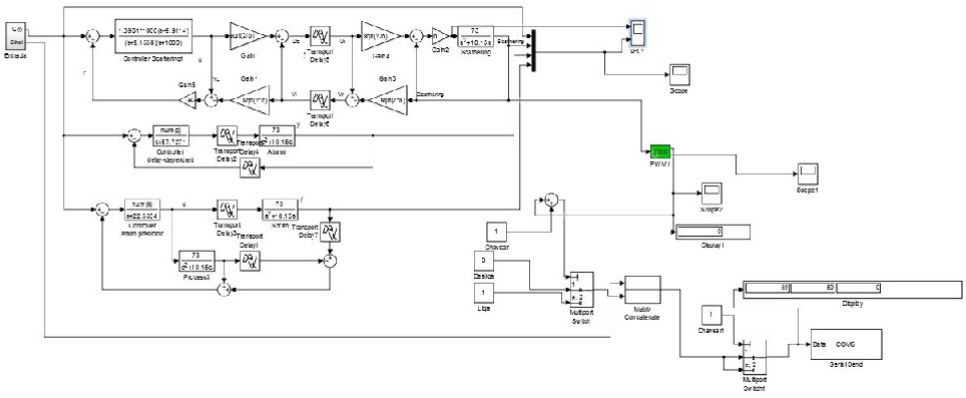

Figura I.3: Modelo completo com PWM assegurado pela Scattering

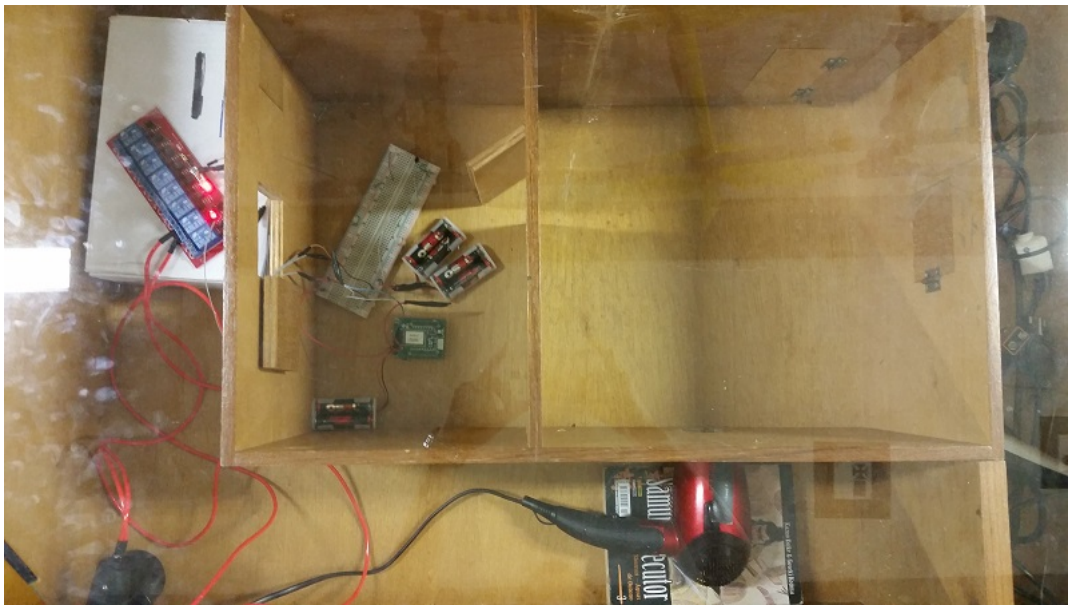

Figura I.4: Imagem do sistema em funcionamento.

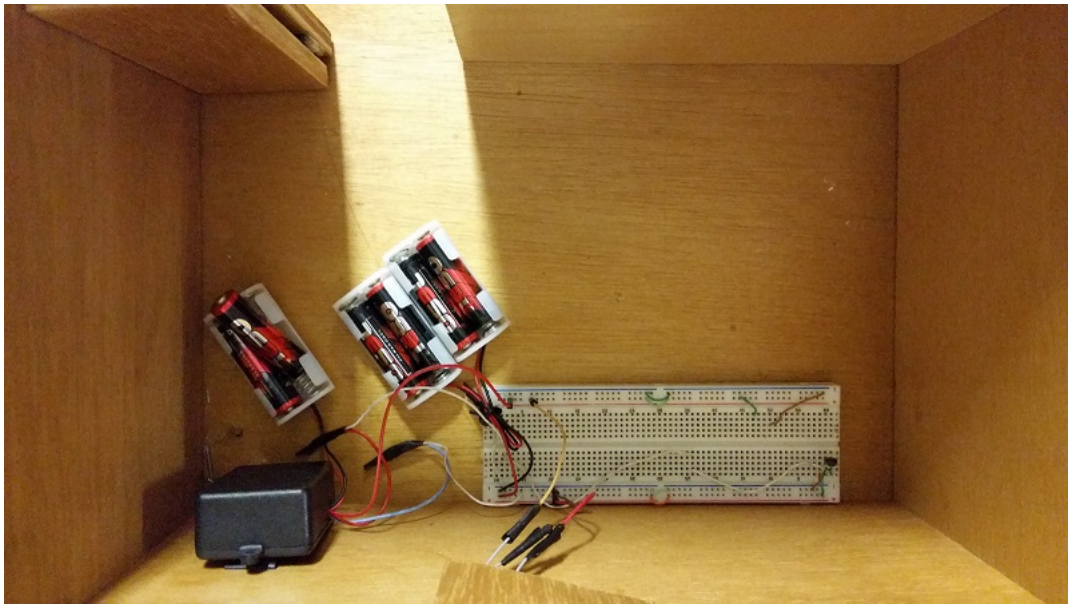

Figura I.5: Imagem do cômodo com o controlador.

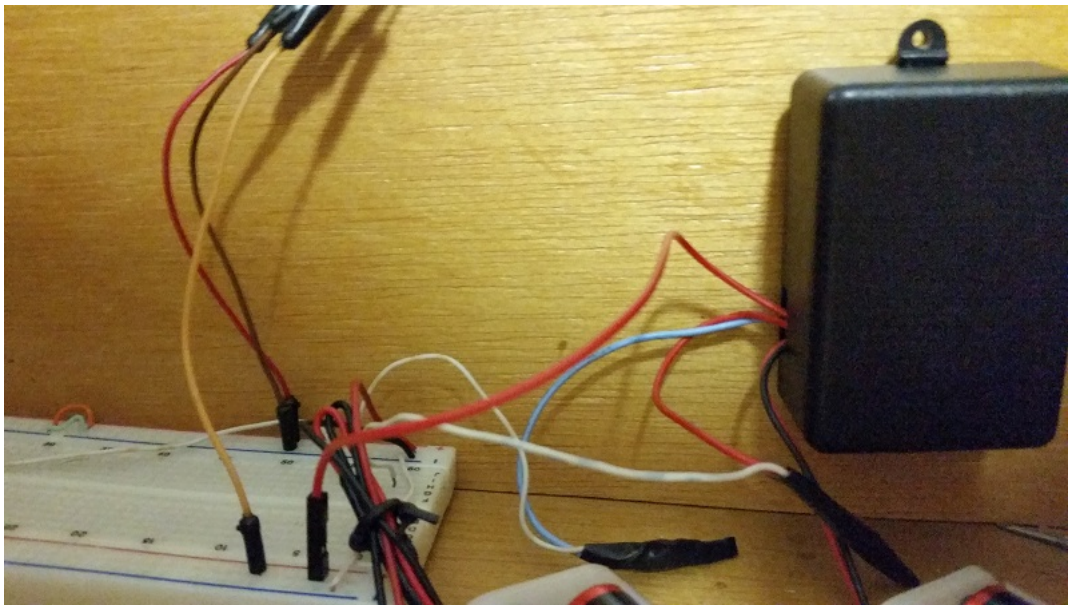

Figura I.6: Vista frontal do módulo periférico.

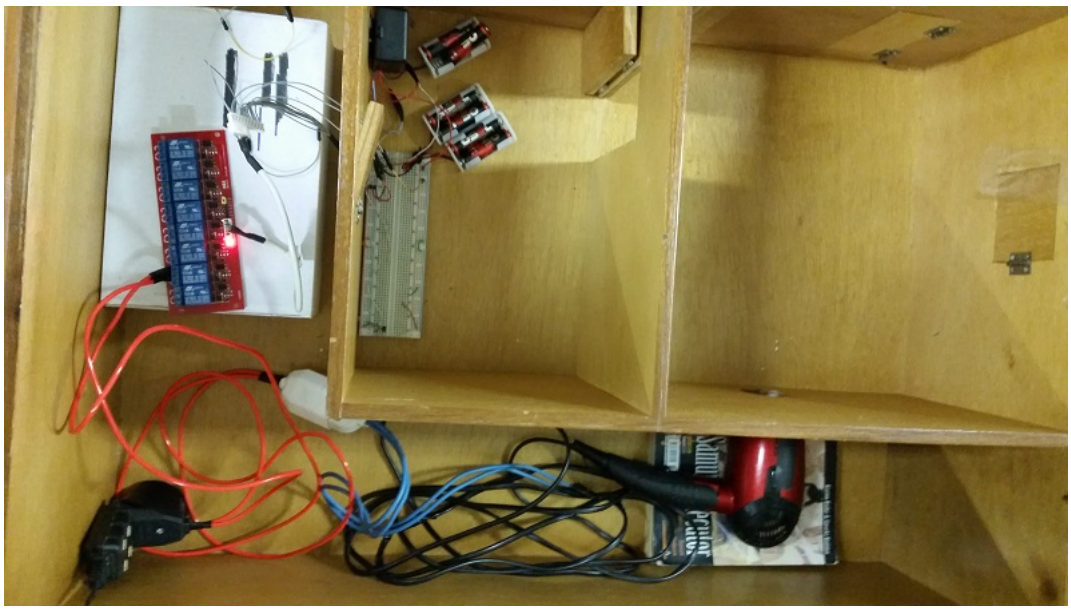

Figura I.7: Vista superior so modelo projetado.

II. DESCRIÇÃO DO CONTEÚDO DO CD

1. Trabalho de Graduação: Transformada de Scatterin aplicada ao controle de sistemas em rede com atrasos variados.
2. Código do Nó Coordenador
3. Código do Nó *End Device*

III. PROGRAMAS UTILIZADOS

Código de programação Nó Coordenador:

/********** Includes section **********/
#include <types.h> #include <taskManager.h> #include <configServer.h> #include <appTimer.h> #include <zdo.h> #include <peer2peer.h> #include <serialInterface.h> #include <bspLeds.h>

#if BSP_ENABLE_RS232_CONTROL == 1 #include "rs232Controller.h" #endif // BSP_ENABLE_RS2

/********** Define(s) section **********/

/********** Variables section **********/
// Network related variables static uint16_t nwkAddr; // Own NWK address static uint16_t nwkAddrdest; static AppMessageBuffer_t appMessageBuffer; // Application message buffer static AppMessageBuffer_t appMessageBuffer2; // Application message buffer static uint8_t messageIdTx = 0; // Application transmitted message ID static uint8_t messageIdRx = 0; // Next expected ID of received message static uint16_t actualDataLength = 0; // Actual data length to be transmitted via network

// Data indications FIFO related variables static uint8_t apsDataIndFifo[APP_DATA_IND_BUFFER_SIZ static uint16_t apsDataIndFifoStart = 0; static uint16_t apsDataIndFifoEnd = 0;

// USART related variables static HAL_UsartDescriptor_t appUsartDescriptor; // USART descriptor (required by stack) static bool usartTxBusy = false; // USART transmission transaction status static uint8_t usartTxBuffer[APP_USART_TX_BUFFER_SIZE]; // USART Tx buffer static uint8_t usartRxBuffer[APP_USART_RX_BUFFER_SIZE]; // USART Rx buffer

// Application related parameters static AppState_t appState = APP_INITIAL_STATE; // application state static AppDataTransmissionState_t appDataTransmissionState; static ZDO_StartNetworkReq networkParams; // request params for ZDO_StartNetworkReq static APS_DataReq_t apsDataReq;

// Endpoint parameters static SimpleDescriptor_t simpleDescriptor = { APP_ENDPOINT, APP_PROFILE_ID, 1, 1, 0, 0 , NULL, 0, NULL }; static APS_RegisterEndpointReq_t endpointParams; static bool noIndications = false;

// Timer indicating starting network during network joining. // Also used as a delay timer between APS_DataConf and APS_DataReq. static HAL_AppTimer_t delayTimer; static HAL_AppTimer_t delayTempo;

#if APP_DETECT_LINK_FAILURE == 1 static uint8_t retryCounter = 0; // Data sen-

ding retries counter // Leave request, used for router to leave the network when communication was interrupted static ZDO_ZdpReq_t leaveReq; #endif // APP_DETECT_LINK_FAILURE

/** Static functions declarations section ********** static void APS_DataIndication(APS_DataInd_t* dataInd); static void APS_DataConf(APS_DataConf_t* confInfo); static void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo); static void initNetwork(void); static void startNetwork(void); static void networkSendData(bool newTransmission);

static void startBlinkTimer(void); static void startingNetworkTimerFired(void); static void delayTimerFired(void);

static void initSerialInterface(void); static void usartStartSending(void); static void usartBytesReceived(uint16_t readBytesLen); static void usartWriteConf(void);

static uint16_t fifoFreeSpace(void); static void fifoWriteData(uint8_t *data, uint16_t size); static uint16_t fifoReadData(uint8_t *data, uint16_t size);

static void buttonReleased(uint8_t buttons);

#if APP_DETECT_LINK_FAILURE == 1 static void leaveNetwork(void); static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp); #endif // APP_DETECT_LINK_FAILURE

/** Implementation section ********** /**********//** \brief Application task handler.

\param none. \return none. ********** void APL_TaskHandler(void) { switch (appState) { case APP_INITIAL_STATE: // Node has initial state initSerialInterface(); // Open USART BSP_OpenButtons(NULL, buttonReleased); // Open buttons BSP_OpenLeds(); // Enable LEDs initNetwork(); SYS_PostTask(APL_TASK_ID); // Execute next step break;

case APP_NETWORK_JOINING_STATE: startBlinkTimer(); startNetwork(); break;

case APP_NETWORK_LEAVING_STATE: case APP_NETWORK_JOINED_STATE: default: break; } }

/**********//** \brief Intializes network parameters.

\param none. \return none. ********** static void initNetwork(void) { DeviceType_t deviceType;

// NWK preconfigured address reading for Config Server CS_ReadParameter(CS_NWK_ADDR_ID, &nwkAddr);

CS_WriteParameter(CS_NWK_UNIQUE_ADDR_ID, &(bool){true});

// Node role detection. If nwkAddr == 0 then node is coordinator // otherwise node role is router. if (0 == nwkAddr) { #ifdef _SECURITY_ { //Set coordinator uid equal to trust center address ExtAddr_t extAddr; CS_ReadParameter(CS_APS_TRUST_CENTER_ADDRESS_ID, &extAddr); CS_WriteParameter(CS_UID_ID, &extAddr); } #endif //_SECURITY_

deviceType = DEVICE_TYPE_COORDINATOR; } else { deviceType = DEVICE_TYPE_ROUTER; }

// Set the deviceType value to Config Server CS_WriteParameter(CS_DEVICE_TYPE_ID, &deviceType);

// Start joining procedure appState = APP_NETWORK_JOINING_STATE; }

/**//** \brief ZDO_StartNetwork primitive confirmation callback.

\param confirmInfo - confirmation parametrs. \return none. ************************************** void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo) { HAL_StopAppTimer(&delayTimer); // Stop the blink timer

if (confirmInfo->status == ZDO_SUCCESS_STATUS) { appDataTransmissionState = APP_DATA_TRA appState = APP_NETWORK_JOINED_STATE;

#if APP_DETECT_LINK_FAILURE == 1 retryCounter = 0; #endif // APP_DETECT_LINK_FAILUR actualDataLength = 0;

//BSP_OnLed(APP_NETWORK_STATUS_LED);

// Setup timer to be used as delay timer between APS_DataConf and APS_DataReq.

// Set application endpoint properties endpointParams.simpleDescriptor = &simpleDescriptor; endpointParams.APS_DataInd = APS_DataIndication; // Register endpoint APS_RegisterEndpointReq(&end

} else {

SYS_PostTask(APL_TASK_ID); } }

/**//** \brief Starts network.

\param none. \return none. ********** static void startNetwork(void) { networkParams.ZDO_StartNetworkConf = ZDO_StartNetworkConf; // start network ZDO_StartNetworkReq(&networkParams); }

#if APP_DETECT_LINK_FAILURE == 1 /********** \brief Leave network.

\param none. \return none. ********** static void leaveNetwork(void) { ZDO_MgmtLeaveReq_t *zdpLeaveReq = &leaveReq.req.reqPayload.mgmtLeav APS_UnregisterEndpointReq_t unregEndpoint;

appState = APP_NETWORK_LEAVING_STATE;

unregEndpoint.endpoint = endpointParams.simpleDescriptor->endpoint; APS_UnregisterEndpointReq(&un

leaveReq.ZDO_ZdpResp = zdpLeaveResp; leaveReq.reqCluster = MGMT_LEAVE_CLID; leaveReq.dstAddrMode = EXT_ADDR_MODE; leaveReq.dstExtAddr = 0; zdpLeaveReq->deviceAddr = 0; zdpLeaveReq->rejoin = 0; zdpLeaveReq->removeChildren = 1; zdpLeaveReq->reserved = 0; ZDO_ZdpReq(&leaveReq); }

/**********//** \brief Leave network response.

\param zdpResp - response data \return none. ********** static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp) { //BSP_OffLed(APP_NETWORK_STATUS_LED);

// Try to rejoin the network appState = APP_NETWORK_JOINING_STATE; SYS_PostTask(APL_TASI

(void)zdpResp; } #endif // APP_DETECT_LINK_FAILURE

/**********//** \brief Starts Blink Timer based on HAL_AppTimer_t.

\param none. \return none. ********** static void startBlinkTimer(void) { delayTimer.interval = APP_JOINING_INDICATION_PERIOD; delayTimer.mode = TIMER_REPEAT_MODE; delayTimer.callback = startingNetworkTimerFired; HAL_StartAppTimer(&delayTimer); }

/**********//** \brief Blink timer callback function.

\param none. \return none. **********

static void startingNetworkTimerFired(void) { //BSP_ToggleLed(APP_NETWORK_STATUS_LED); }

/**********//** \brief Update network status event handler.

\param nwkParams - new network parameters. \return none. ********** void ZDO_MgmtNwkUpdateNotf(ZDO_MgmtNwkUpdateNotf_t *nwkParams) { switch (nwkParams->status) { case ZDO_NETWORK_STARTED_STATUS: break;

case ZDO_NETWORK_LOST_STATUS: { APS_UnregisterEndpointReq_t unregEndpoint;

unregEndpoint.endpoint = endpointParams.simpleDescriptor->endpoint; APS_UnregisterEndpointReq(&un

//BSP_OffLed(APP_NETWORK_STATUS_LED);

// try to rejoin the network appState = APP_NETWORK_JOINING_STATE; SYS_PostTask(APL_TASK break; }

case ZDO_NWK_UPDATE_STATUS: break;

default: break; } }

/**********//** \brief

Wakeup event handler (dummy).

\param none. \return none. **********
void ZDO_WakeUpInd(void) { }

/**********//** \brief
Sends data to the network.

\param newTransmission - new data transmission or retransmission

\return none. **********/
static void networkSendData(bool newTransmission) { if (APP_DATA_TRANSMISSION_SENDING_STATE
== appDataTransmissionState) { appDataTransmissionState = APP_DATA_TRANSMISSION_BUSY_STAT
// indicate we're sending //BSP_OnLed(APP_SENDING_STATUS_LED);

appMessageBuffer.message.data[2]=nwkAddrdest; //Adequar a comunicacao do matlab ao BacNet appMessageBuffer.message.data[3]=nwkAddr; //Adequar a comunicacao do matlab ao BacNet appMessageBuffer.message.data[8]=appMessageBuffer.message.data[11]; //Comando do matlab

if (newTransmission) { // The new applicatiom message will be sent // messageId value appMessageBuffer.message.messageId = messageIdTx++; // APS Data Request preparing apsDataReq.dstAddrMode = APS_SHORT_ADDRESS; // Short addressing mode // Destination address apsDataReq.dstAddress.shortAddress = nwkAddrdest; apsDataReq.profileId = simpleDescriptor.AppProfileId; // Profile ID apsDataReq.dstEndpoint = simpleDescriptor.endpoint; // Desctination endpoint apsDataReq.clusterId = APP_CLUSTER_ID; // Desctination cluster ID apsDataReq.srcEndpoint = simpleDescriptor.endpoint; // Source endpoint apsDataReq.asdu = (uint8_t*) &appMessageBuffer.message; // application message pointer // actual application message length apsDataReq.asduLength = actualDataLength + sizeof(appMessageBuffer.message.messageId); apsDataReq.txOptions.acknowledgedTransmission = 1; // Acknowledged transmission enabled #if APP_FRAGMENTATION apsDataReq.txOptions.fragmentationPermitted = 1; #else apsDataReq.txOptions.fragmentationPermitted = 0; #endif // APP_FRAGMENTATION apsDataReq.radius = 0; // Use maximal possible radius apsDataReq.APS_DataConf = APS_DataConf; // Confirm handler }

APS_DataReq(&apsDataReq); } }

/**********//** \brief
Handler of aps data sent confirmation.

\param confInfo - confirmation info \return none. **********
void APS_DataConf(APS_DataConf_t* confInfo) { //BSP_OffLed(APP_SENDING_STATUS_LED);

if (APS_SUCCESS_STATUS != confInfo->status) { #if APP_DETECT_LINK_FAILURE == 1 retryCounter++; if (MAX_RETRIES_BEFORE_REJOIN == retryCounter) { leaveNetwork(); } else #endif // APP_DETECT_LINK_FAILURE { // Data not delivered, resend. appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; networkSendData(false); } return; }

#if APP_DETECT_LINK_FAILURE == 1 retryCounter = 0; #endif // APP_DETECT_LINK_FAILUR

actualDataLength = 0;

#if APP_TRANSMISSION_DELAY > 0 appDataTransmissionState = APP_DATA_TRANSMISSION_S

delayTimer.interval = APP_TRANSMISSION_DELAY; delayTimer.mode = TIMER_ONE_SHOT_MOD
delayTimer.callback = delayTimerFired; HAL_StartAppTimer(&delayTimer); #else appData-
TransmissionState = APP_DATA_TRANSMISSION_READY_STATE; usartBytesReceived(0);
#endif }

/**//** \brief
Delay timer callback function.

\param none. \return none. **
static void delayTimerFired(void) { if (APP_DATA_TRANSMISSION_STOP_STATE == app-
DataTransmissionState) appDataTransmissionState = APP_DATA_TRANSMISSION_READY_STATE;
else appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; usartBy-
tesReceived(0); }

/**//** \brief
APS data indication handler.

\param indData - received data pointer. \return none. **
void APS_DataIndication(APS_DataInd_t* indData) { AppMessage_t *appMessage = (App-
Message_t *) indData->asdu;

// Data received indication //BSP_OnLed(APP_RECEIVING_STATUS_LED);

if (appMessage->messageId == messageIdRx) { fifoWriteData(appMessage->data, indData-
>asduLength - 1);

usartStartSending();

if (fifoFreeSpace() < APP_APS_PAYLOAD_SIZE) { APS_StopEndpointIndication(APP_ENDPOINT);
// Disable APS Data Indications noIndications = true; } }

messageIdRx = appMessage->messageId + 1;

//BSP_OffLed(APP_RECEIVING_STATUS_LED); }

/**//** \brief
Initializes USART or VCP (depends on user's settings).

\param none. \return none. **
static void initSerialInterface(void) { usartTxBusy = false;

appUsartDescriptor.tty = USART_CHANNEL; appUsartDescriptor.mode = USART_MODE_ASYNC;
appUsartDescriptor.baudrate = USART_BAUDRATE_38400; appUsartDescriptor.dataLength =
USART_DATA8; appUsartDescriptor.parity = USART_PARITY_NONE; appUsartDescriptor.stopbits
= USART_STOPBIT_1; appUsartDescriptor.rxBuffer = usartRxBuffer; appUsartDescriptor.rxBufferLength
= sizeof(usartRxBuffer); appUsartDescriptor.txBuffer = NULL; // use callback mode appUsart-
Descriptor.txBufferLength = 0; appUsartDescriptor.rxCallback = usartBytesReceived; appUsart-

Descriptor.txCallback = usartWriteConf; appUsartDescriptor.flowControl = USART_FLOW_CONTROL_HA

#if BSP_ENABLE_RS232_CONTROL == 1 BSP_EnableRs232(); #endif // BSP_ENABLE_RS232_C

OPEN_USART(&appUsartDescriptor); }

/**//** \brief
Initiates data transmission via serial interface.

\param none \return none **********
static void usartStartSending(void) { uint16_t size;

if (!usartTxBusy) { size = fifoReadData(usartTxBuffer, APP_USART_TX_BUFFER_SIZE);
WRITE_USART(&appUsartDescriptor, usartTxBuffer, size);

usartTxBusy = true;

if (noIndications && (fifoFreeSpace() >= APP_APS_PAYLOAD_SIZE)) { APS_ResumeEndpointIndicatio
// Enable APS Data Indications noIndications = false; } } }

/**//** \brief
Writing confirmation has been received. New message can be sent.

\param none \return none **********
static void usartWriteConf(void) { usartTxBusy = false;

if (APP_DATA_IND_BUFFER_SIZE != fifoFreeSpace()) usartStartSending(); }

/**//** \brief
New USART bytes are received - HAL USART callback function.

\param bytesToRead - received bytes amount. \return none. **********
static void usartBytesReceived(uint16_t bytesToRead) { if (APP_NETWORK_JOINED_STATE
!= appState) return; if ((APP_DATA_TRANSMISSION_BUSY_STATE == appDataTransmis-
sionState) || (APP_DATA_TRANSMISSION_STOP_STATE == appDataTransmissionState))
return;

if (actualDataLength < APP_MAX_PACKET_SIZE) { actualDataLength += (uint8_t)
READ_USART(&appUsartDescriptor, appMessageBuffer.message.data + actualDataLength, APP_MAX_PAC
- actualDataLength); }

#if APP_DELAY_BEFORE_SEND > 0 if (actualDataLength < APP_MAX_PACKET_SIZE)
{ if ((APP_DATA_TRANSMISSION_READY_STATE == appDataTransmissionState) && (ac-
tualDataLength > 0)) { appDataTransmissionState = APP_DATA_TRANSMISSION_WAIT_STATE;
delayTimer.interval = APP_DELAY_BEFORE_SEND; delayTimer.mode = TIMER_ONE_SHOT_MODE;
delayTimer.callback = delayTimerFired; HAL_StartAppTimer(&delayTimer); return; } if (APP_DATA_TRAN
== appDataTransmissionState) return; } else if (APP_DATA_TRANSMISSION_WAIT_STATE
== appDataTransmissionState) HAL_StopAppTimer(&delayTimer); #endif

if (actualDataLength > 0) { appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STA
networkSendData(true); }

(void)bytesToRead; }

/**//** \brief
Get free space size in data indications FIFO

\param none \return Free space size in FIFO **
static uint16_t fifoFreeSpace(void) { int16_t free = apsDataIndFifoStart - apsDataIndFifoEnd;

if (apsDataIndFifoStart <= apsDataIndFifoEnd) free += sizeof(apsDataIndFifo);

return free; }

/**//** \brief
Write data into data indications FIFO. FIFO must have enough space to fit all data.

\param data - pointer to the data size - size of provided data (bytes) \return none ************************
static void fifoWriteData(uint8_t *data, uint16_t size) { uint16_t i;

for (i = 0; i < size; i++) { apsDataIndFifo[apsDataIndFifoEnd++] = data[i]; if (apsDataInd-
FifoEnd == sizeof(apsDataIndFifo)) apsDataIndFifoEnd = 0; } }

/**//** \brief
Read data from data indications FIFO

\param data - pointer to the buffer where data should be placed size - maximum size of data to
read \return Actual amount of data read **
static uint16_t fifoReadData(uint8_t *data, uint16_t size) { uint16_t read = 0; uint16_t i;
BSP_OpenLeds();

for (i = 0; i < size; i++) { if (apsDataIndFifoStart == apsDataIndFifoEnd) break;

appMessageBuffer2.message.data[i]=apsDataIndFifo[apsDataIndFifoStart++]; if (i==4) { nw-
kAddrdest=(appMessageBuffer2.message.data[i]); } apsDataIndFifoStart=apsDataIndFifoStart−;

data[i] = apsDataIndFifo[apsDataIndFifoStart++]; read++;

if (apsDataIndFifoStart == sizeof(apsDataIndFifo)) apsDataIndFifoStart = 0; }

return read; }

/**//** \brief
Button released event handler.

\param aButton - released button number \return none. **
static void buttonReleased(uint8_t aButton) { switch (aButton) { case BSP_KEY0: // SW1 pres-
sed if ((APP_NETWORK_JOINED_STATE == appState) && (APP_DATA_TRANSMISSION_READY_S
== appDataTransmissionState)) { // Fill the buffer with some data and send it memset(&appMessageBuffer.mess
0xdd, APP_MAX_PACKET_SIZE); actualDataLength = APP_MAX_PACKET_SIZE; app-
DataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; networkSendData(true);
break; }

default: break; } } #ifdef _BINDING_ /***
Stub for ZDO Binding Indication

Parameters: bindInd - indication

Return: none

**********/
void ZDO_BindIndication(ZDO_BindInd_t *bindInd) { (void)bindInd; }

/**********
Stub for ZDO Unbinding Indication

Parameters: unbindInd - indication

Return: none

**********/
void ZDO_UnbindIndication(ZDO_UnbindInd_t *unbindInd) { (void)unbindInd; } #endif //_BINDING_

/**********//** \brief Main - C program main start function

\param none \return none **********
int main(void) { SYS_SysInit();

for(;;) { SYS_RunTask(); } } // eof peer2Peer.c

Código de programação Nó *End Device*:

/********** Includes section **********/
#include <types.h> #include <taskManager.h> #include <configServer.h> #include <appTimer.h> #include <zdo.h> #include <peer2peer.h> #include <serialInterface.h> #include <bspLeds.h>

#if BSP_ENABLE_RS232_CONTROL == 1 #include "rs232Controller.h" #endif // BSP_ENABLE_RS2

/********** Define(s) section **********/

/********** Variables section **********/
// Network related variables static uint16_t nwkAddr; // Own NWK address static int8_t valor; // Valor do sensor de temperatura static int8_t flag=0; static int8_t flag2=0; static int8_t contador=0; static AppMessageBuffer_t appMessageBuffer; // Application message buffer static AppMessageBuffer_t appMessageBuffer2; // Application message buffer static uint8_t messageIdTx = 0; // Application transmitted message ID static uint8_t messageIdRx = 0; // Next expected ID of received message static uint16_t actualDataLength = 0; // Actual data length to be transmitted via network

// Data indications FIFO related variables static uint8_t apsDataIndFifo[APP_DATA_IND_BUFFER_SIZ static uint16_t apsDataIndFifoStart = 0; static uint16_t apsDataIndFifoEnd = 0;

// USART related variables static HAL_UsartDescriptor_t appUsartDescriptor; // USART descriptor (required by stack) static bool usartTxBusy = false; // USART transmission transaction status static uint8_t usartTxBuffer[APP_USART_TX_BUFFER_SIZE]; // USART Tx buffer static uint8_t usartRxBuffer[APP_USART_RX_BUFFER_SIZE]; // USART Rx buffer

// Application related parameters static AppState_t appState = APP_INITIAL_STATE; // application state static AppDataTransmissionState_t appDataTransmissionState; static ZDO_StartNetworkReq networkParams; // request params for ZDO_StartNetworkReq static APS_DataReq_t apsDataReq;

// Endpoint parameters static SimpleDescriptor_t simpleDescriptor = { APP_ENDPOINT, APP_PROFILE_ID, 1, 1, 0, 0 , NULL, 0, NULL }; static APS_RegisterEndpointReq_t endpointParams; static bool noIndications = false;

// Timer indicating starting network during network joining. // Also used as a delay timer between APS_DataConf and APS_DataReq. static HAL_AppTimer_t delayTimer; static HAL_AppTimer_t delayTempo;

#if APP_DETECT_LINK_FAILURE == 1 static uint8_t retryCounter = 0; // Data sending retries counter // Leave request, used for router to leave the network when communication was interrupted static ZDO_ZdpReq_t leaveReq; #endif // APP_DETECT_LINK_FAILURE

/**********

Static functions declarations section ********** static void APS_DataIndication(APS_DataInd_t* dataInd); static void APS_DataConf(APS_DataConf_t* confInfo); static void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo); static void initNetwork(void); static void startNetwork(void); static void networkSendData(bool newTransmission);

static void startBlinkTimer(void); static void startingNetworkTimerFired(void); static void timerEnvioTemperaturaCB(void); static void delayTimerFired(void);

static void initSerialInterface(void); static void usartStartSending(void); static void usartBytesReceived(uint16_t readBytesLen); static void usartWriteConf(void);

static uint16_t fifoFreeSpace(void); static void fifoWriteData(uint8_t *data, uint16_t size); static uint16_t fifoReadData(uint8_t *data, uint16_t size);

static void buttonReleased(uint8_t buttons);

#if APP_DETECT_LINK_FAILURE == 1 static void leaveNetwork(void); static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp); #endif // APP_DETECT_LINK_FAILURE

/********** Implementation section ********** /**********//** \brief Application task handler.

\param none. \return none. ********** void APL_TaskHandler(void) { switch (appState) { case APP_INITIAL_STATE: // Node has initial state initSerialInterface(); // Open USART BSP_OpenButtons(NULL, buttonReleased); // Open buttons BSP_OpenLeds(); // Enable LEDs initNetwork(); SYS_PostTask(APL_TASK_ID); // Execute next step break;

case APP_NETWORK_JOINING_STATE: startBlinkTimer(); startNetwork(); break;

case APP_NETWORK_LEAVING_STATE: case APP_NETWORK_JOINED_STATE: default: break; } }

/**********//** \brief Intializes network parameters.

\param none. \return none. ********** static void initNetwork(void) { DeviceType_t deviceType;

// NWK preconfigured address reading for Config Server CS_ReadParameter(CS_NWK_ADDR_ID, &nwkAddr);

CS_WriteParameter(CS_NWK_UNIQUE_ADDR_ID, &(bool){true});

// Node role detection. If nwkAddr == 0 then node is coordinator // otherwise node role is router. if (0 == nwkAddr) { #ifdef _SECURITY_ { //Set coordinator uid equal to trust center address ExtAddr_t extAddr; CS_ReadParameter(CS_APS_TRUST_CENTER_ADDRESS_ID, &extAddr); CS_WriteParameter(CS_UID_ID, &extAddr); } #endif //_SECURITY_

deviceType = DEVICE_TYPE_COORDINATOR; } else { deviceType = DEVICE_TYPE_ROUTER; }

// Set the deviceType value to Config Server CS_WriteParameter(CS_DEVICE_TYPE_ID, &deviceType);

// Start joining procedure appState = APP_NETWORK_JOINING_STATE; }

/**********//** \brief ZDO_StartNetwork primitive confirmation callback.

\param confirmInfo - confirmation parametrs. \return none. ********** void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo) { HAL_StopAppTimer(&delayTimer); // Stop the blink timer

if (confirmInfo->status == ZDO_SUCCESS_STATUS) { appDataTransmissionState = APP_DATA_TRA appState = APP_NETWORK_JOINED_STATE;

#if APP_DETECT_LINK_FAILURE == 1 retryCounter = 0; #endif // APP_DETECT_LINK_FAILUR actualDataLength = 0;

//BSP_OnLed(APP_NETWORK_STATUS_LED);

// Setup timer to be used as delay timer between APS_DataConf and APS_DataReq.

// Set application endpoint properties endpointParams.simpleDescriptor = &simpleDescriptor; endpointParams.APS_DataInd = APS_DataIndication; // Register endpoint APS_RegisterEndpointReq(&end

delayTempo.interval = 1000L; //mudar para 1000L delayTempo.mode = TIMER_REPEAT_MODE; delayTempo.callback = timerEnvioTemperaturaCB; HAL_StartAppTimer(&delayTempo); } else {

SYS_PostTask(APL_TASK_ID); } }

/**********//** \brief Starts network.

\param none. \return none. ********** static void startNetwork(void) { networkParams.ZDO_StartNetworkConf = ZDO_StartNetworkConf; // start network ZDO_StartNetworkReq(&networkParams); }

#if APP_DETECT_LINK_FAILURE == 1 /********** \brief Leave network.

\param none. \return none. ********** static void leaveNetwork(void) { ZDO_MgmtLeaveReq_t *zdpLeaveReq = &leaveReq.req.reqPayload.mgmtLeav APS_UnregisterEndpointReq_t unregEndpoint;

appState = APP_NETWORK_LEAVING_STATE;

unregEndpoint.endpoint = endpointParams.simpleDescriptor->endpoint; APS_UnregisterEndpointReq(&un

leaveReq.ZDO_ZdpResp = zdpLeaveResp; leaveReq.reqCluster = MGMT_LEAVE_CLID; le-

aveReq.dstAddrMode = EXT_ADDR_MODE; leaveReq.dstExtAddr = 0; zdpLeaveReq->deviceAddr = 0; zdpLeaveReq->rejoin = 0; zdpLeaveReq->removeChildren = 1; zdpLeaveReq->reserved = 0; ZDO_ZdpReq(&leaveReq); }

/***//** \brief Leave network response.

\param zdpResp - response data \return none. ********** static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp) { //BSP_OffLed(APP_NETWORK_STATUS_LED);

// Try to rejoin the network appState = APP_NETWORK_JOINING_STATE; SYS_PostTask(APL_TASK

(void)zdpResp; } #endif // APP_DETECT_LINK_FAILURE

/***//** \brief Starts Blink Timer based on HAL_AppTimer_t.

\param none. \return none. ********** static void startBlinkTimer(void) { delayTimer.interval = APP_JOINING_INDICATION_PERIOD; delayTimer.mode = TIMER_REPEAT_MODE; delayTimer.callback = startingNetworkTimerFired; HAL_StartAppTimer(&delayTimer); }

/***//** \brief Blink timer callback function.

\param none. \return none. ********** static void temperaturesSensorHandler(uint8_t batV) {

uint8_t batV2; batV2=batV-60; uint8_t temperatura; temperatura=(((batV2-20)*0.165)+21+(flag*46)); //conversao da temperatura //segue trecho de codigo para leitura da temperatura em qualquer valor if (flag2>0) { if((temperatura-valor)<(-25)){ flag=flag+1; temperatura=(((batV2-20)*0.165)+21+(flag*46)); }else if ((temperatura-valor)>30) { flag=flag-1; temperatura=(((batV2-20)*0.165)+21+(flag*46)); } }else{ flag2=flag2+1; } //estrutura da mensagem no formato bacnet appMessageBuffer.message.data[0]=55; //Preâmbulo do BacNet appMessageBuffer.message.data[1]=255; //Preâmbulo do BacNet appMessageBuffer.message.data[2]=6; //Tipo do frame appMessageBuffer.message.data[3]=0; //Endereco de destino appMessageBuffer.message.data[4]=nwkAddr; //Endereco fonte appMessageBuffer.message.data[5]=11; //tamanho appMessageBuffer.message.data[6]=11; //tamanho appMessageBuffer.message.data[7]=1; //CRC appMessageBuffer.message.data[8]=temperatura; //Dados appMessageBuffer.message.data[9]=1; //CRC appMessageBuffer.message.data[10]=1; //CRC

valor=temperatura; //flag de controle usado na infericao da temperatura

appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; actualDataLength=11; networkSendData(true);

}

static void startingNetworkTimerFired(void) { //BSP_ToggleLed(APP_NETWORK_STATUS_LED); }

static void timerEnvioTemperaturaCB(void) { BSP_OpenBatterySensor(); BSP_ReadBatteryData(tempera

}

/**//** \brief
Update network status event handler.

\param nwkParams - new network parameters. \return none. ***********************************
void ZDO_MgmtNwkUpdateNotf(ZDO_MgmtNwkUpdateNotf_t *nwkParams) { switch (nwkParams->status) { case ZDO_NETWORK_STARTED_STATUS: break;

case ZDO_NETWORK_LOST_STATUS: { APS_UnregisterEndpointReq_t unregEndpoint;

unregEndpoint.endpoint = endpointParams.simpleDescriptor->endpoint; APS_UnregisterEndpointReq(&un

//BSP_OffLed(APP_NETWORK_STATUS_LED);

// try to rejoin the network appState = APP_NETWORK_JOINING_STATE; SYS_PostTask(APL_TASK break; }

case ZDO_NWK_UPDATE_STATUS: break;

default: break; } }

/**//** \brief
Wakeup event handler (dummy).

\param none. \return none. ***********************************
void ZDO_WakeUpInd(void) { }

/**//** \brief
Sends data to the network.

\param newTransmission - new data transmission or retransmission

\return none. ***********************************/
static void networkSendData(bool newTransmission) { if (APP_DATA_TRANSMISSION_SENDING_STATE == appDataTransmissionState) { appDataTransmissionState = APP_DATA_TRANSMISSION_BUSY_STAT // indicate we're sending //BSP_OnLed(APP_SENDING_STATUS_LED);

if (newTransmission) { // The new applicatiom message will be sent // messageId value appMessageBuffer.message.messageId = messageIdTx++; // APS Data Request preparing apsDataReq.dstAddrMode = APS_SHORT_ADDRESS; // Short addressing mode // Destination address apsDataReq.dstAddress.shortAddress = 0; apsDataReq.profileId = simpleDescriptor.AppProfileId; // Profile ID apsDataReq.dstEndpoint = simpleDescriptor.endpoint; // Desctination endpoint apsDataReq.clusterId = APP_CLUSTER_ID; // Desctination cluster ID apsDataReq.srcEndpoint = simpleDescriptor.endpoint; // Source endpoint apsDataReq.asdu = (uint8_t*) &appMessageBuffer.message; // application message pointer // actual application message length apsDataReq.asduLength = actualDataLength + sizeof(appMessageBuffer.message.messageId); apsDataReq.txOptions.acknowledgedTransmission = 1; // Acknowledged transmission enabled #if APP_FRAGMENTAT apsDataReq.txOptions.fragmentationPermitted = 1; #else apsDataReq.txOptions.fragmentationPermitted = 0; #endif // APP_FRAGMENTATION apsDataReq.radius = 0; // Use maximal possible radius apsDataReq.APS_DataConf = APS_DataConf; // Confirm handler }

APS_DataReq(&apsDataReq); } }

/**//** \brief Handler of aps data sent confirmation.

\param confInfo - confirmation info \return none. ********** void APS_DataConf(APS_DataConf_t* confInfo) { //BSP_OffLed(APP_SENDING_STATUS_LED);

if (APS_SUCCESS_STATUS != confInfo->status) { #if APP_DETECT_LINK_FAILURE == 1 retryCounter++; if (MAX_RETRIES_BEFORE_REJOIN == retryCounter) { leaveNetwork(); } else #endif // APP_DETECT_LINK_FAILURE { // Data not delivered, resend. appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; networkSendData(false); } return; }

#if APP_DETECT_LINK_FAILURE == 1 retryCounter = 0; #endif // APP_DETECT_LINK_FAILUR actualDataLength = 0;

#if APP_TRANSMISSION_DELAY > 0 appDataTransmissionState = APP_DATA_TRANSMISSION_S

delayTimer.interval = APP_TRANSMISSION_DELAY; delayTimer.mode = TIMER_ONE_SHOT_MOD delayTimer.callback = delayTimerFired; HAL_StartAppTimer(&delayTimer); #else appDataTransmissionState = APP_DATA_TRANSMISSION_READY_STATE; usartBytesReceived(0); #endif }

/**//** \brief Delay timer callback function.

\param none. \return none. ********** static void delayTimerFired(void) { if (APP_DATA_TRANSMISSION_STOP_STATE == appDataTransmissionState) appDataTransmissionState = APP_DATA_TRANSMISSION_READY_STATE; else appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; usartBytesReceived(0); }

/**//** \brief APS data indication handler.

\param indData - received data pointer. \return none. ********** void APS_DataIndication(APS_DataInd_t* indData) { AppMessage_t *appMessage = (AppMessage_t *) indData->asdu;

// Data received indication //BSP_OnLed(APP_RECEIVING_STATUS_LED);

if (appMessage->messageId == messageIdRx) { fifoWriteData(appMessage->data, indData->asduLength - 1);

usartStartSending();

if (fifoFreeSpace() < APP_APS_PAYLOAD_SIZE) { APS_StopEndpointIndication(APP_ENDPOINT); // Disable APS Data Indications noIndications = true; } }

messageIdRx = appMessage->messageId + 1;

//BSP_OffLed(APP_RECEIVING_STATUS_LED); }

/***//** \brief Initializes USART or VCP (depends on user's settings).

\param none. \return none. ********** static void initSerialInterface(void) { usartTxBusy = false;

appUsartDescriptor.tty = USART_CHANNEL; appUsartDescriptor.mode = USART_MODE_ASYNC; appUsartDescriptor.baudrate = USART_BAUDRATE_38400; appUsartDescriptor.dataLength = USART_DATA8; appUsartDescriptor.parity = USART_PARITY_NONE; appUsartDescriptor.stopbits = USART_STOPBIT_1; appUsartDescriptor.rxBuffer = usartRxBuffer; appUsartDescriptor.rxBufferLength = sizeof(usartRxBuffer); appUsartDescriptor.txBuffer = NULL; // use callback mode appUsartDescriptor.txBufferLength = 0; appUsartDescriptor.rxCallback = usartBytesReceived; appUsartDescriptor.txCallback = usartWriteConf; appUsartDescriptor.flowControl = USART_FLOW_CONTROL_HA

#if BSP_ENABLE_RS232_CONTROL == 1 BSP_EnableRs232(); #endif // BSP_ENABLE_RS232_C

OPEN_USART(&appUsartDescriptor); }

/***//** \brief Initiates data transmission via serial interface.

\param none \return none ********** static void usartStartSending(void) { uint16_t size;

if (!usartTxBusy) { size = fifoReadData(usartTxBuffer, APP_USART_TX_BUFFER_SIZE); WRITE_USART(&appUsartDescriptor, usartTxBuffer, size);

usartTxBusy = true;

if (noIndications && (fifoFreeSpace() >= APP_APS_PAYLOAD_SIZE)) { APS_ResumeEndpointIndicatio // Enable APS Data Indications noIndications = false; } } }

/***//** \brief Writing confirmation has been received. New message can be sent.

\param none \return none ********** static void usartWriteConf(void) { usartTxBusy = false;

if (APP_DATA_IND_BUFFER_SIZE != fifoFreeSpace()) usartStartSending(); }

/***//** \brief New USART bytes are received - HAL USART callback function.

\param bytesToRead - received bytes amount. \return none. ********** static void usartBytesReceived(uint16_t bytesToRead) { if (APP_NETWORK_JOINED_STATE != appState) return; if ((APP_DATA_TRANSMISSION_BUSY_STATE == appDataTransmissionState) || (APP_DATA_TRANSMISSION_STOP_STATE == appDataTransmissionState)) return;

if (actualDataLength < APP_MAX_PACKET_SIZE) { actualDataLength += (uint8_t)

READ_USART(&appUsartDescriptor, appMessageBuffer.message.data + actualDataLength, APP_MAX_PAC
- actualDataLength); }

#if APP_DELAY_BEFORE_SEND > 0 if (actualDataLength < APP_MAX_PACKET_SIZE)
{ if ((APP_DATA_TRANSMISSION_READY_STATE == appDataTransmissionState) && (ac-
tualDataLength > 0)) { appDataTransmissionState = APP_DATA_TRANSMISSION_WAIT_STATE;
delayTimer.interval = APP_DELAY_BEFORE_SEND; delayTimer.mode = TIMER_ONE_SHOT_MODE;
delayTimer.callback = delayTimerFired; HAL_StartAppTimer(&delayTimer); return; } if (APP_DATA_TRAN
== appDataTransmissionState) return; } else if (APP_DATA_TRANSMISSION_WAIT_STATE
== appDataTransmissionState) HAL_StopAppTimer(&delayTimer); #endif

if (actualDataLength > 0) { appDataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STA
networkSendData(true); }

(void)bytesToRead; }

/**//** \brief
Get free space size in data indications FIFO

\param none \return Free space size in FIFO **
static uint16_t fifoFreeSpace(void) { int16_t free = apsDataIndFifoStart - apsDataIndFifoEnd;

if (apsDataIndFifoStart <= apsDataIndFifoEnd) free += sizeof(apsDataIndFifo);

return free; }

/**//** \brief
Write data into data indications FIFO. FIFO must have enough space to fit all data.

\param data - pointer to the data size - size of provided data (bytes) \return none **********************
static void fifoWriteData(uint8_t *data, uint16_t size) { uint16_t i;

for (i = 0; i < size; i++) { apsDataIndFifo[apsDataIndFifoEnd++] = data[i]; if (apsDataInd-
FifoEnd == sizeof(apsDataIndFifo)) apsDataIndFifoEnd = 0; } }

/**//** \brief
Read data from data indications FIFO

\param data - pointer to the buffer where data should be placed size - maximum size of data to
read \return Actual amount of data read **
static uint16_t fifoReadData(uint8_t *data, uint16_t size) { uint16_t read = 0; uint16_t i;
BSP_OpenLeds();

for (i = 0; i < size; i++) { if (apsDataIndFifoStart == apsDataIndFifoEnd) break;

appMessageBuffer2.message.data[i]=apsDataIndFifo[apsDataIndFifoStart++]; if (i==7) { if
((appMessageBuffer2.message.data[i])==1) {

halOnRedLed(); }else if ((appMessageBuffer2.message.data[i])==0) {

halOffRedLed(); } }

apsDataIndFifoStart=apsDataIndFifoStart−;

data[i] = apsDataIndFifo[apsDataIndFifoStart++]; read++;

if (apsDataIndFifoStart == sizeof(apsDataIndFifo)) apsDataIndFifoStart = 0; }

return read; }

/**//** \brief
Button released event handler.

\param aButton - released button number \return none. **
static void buttonReleased(uint8_t aButton) { switch (aButton) { case BSP_KEY0: // SW1 pres-
sed if ((APP_NETWORK_JOINED_STATE == appState) && (APP_DATA_TRANSMISSION_READY_S
== appDataTransmissionState)) { // Fill the buffer with some data and send it memset(&appMessageBuffer.mess
0xdd, APP_MAX_PACKET_SIZE); actualDataLength = APP_MAX_PACKET_SIZE; app-
DataTransmissionState = APP_DATA_TRANSMISSION_SENDING_STATE; networkSendData(true);
break; }

default: break; } } #ifdef _BINDING_ /***
Stub for ZDO Binding Indication

Parameters: bindInd - indication

Return: none

***/
void ZDO_BindIndication(ZDO_BindInd_t *bindInd) { (void)bindInd; }

/***
Stub for ZDO Unbinding Indication

Parameters: unbindInd - indication

Return: none

***/
void ZDO_UnbindIndication(ZDO_UnbindInd_t *unbindInd) { (void)unbindInd; } #endif //_BIN-
DING_

/**//** \brief Main
- C program main start function

\param none \return none **
int main(void) { SYS_SysInit();

for(;;) { SYS_RunTask(); } } // eof peer2Peer.c