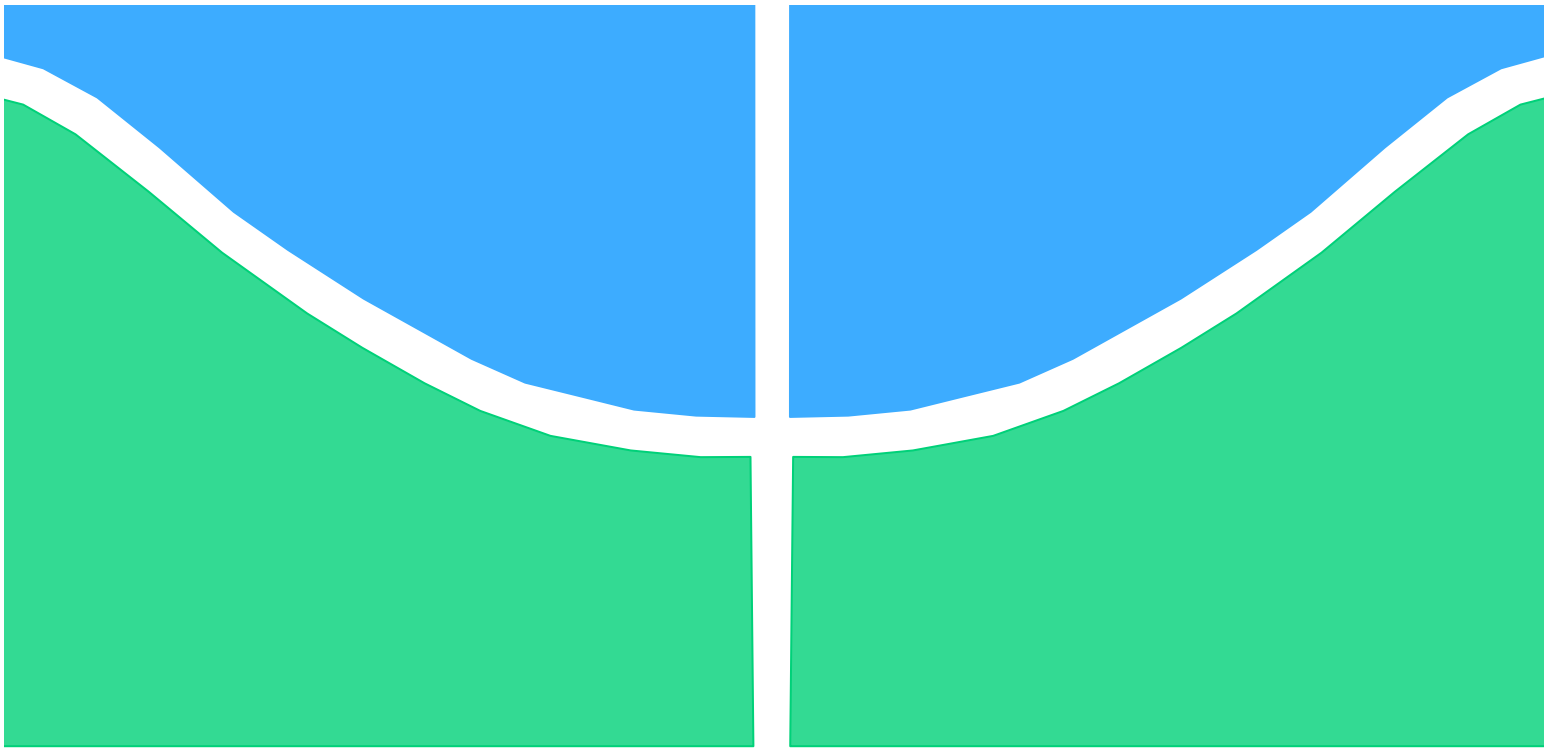


TRABALHO DE GRADUAÇÃO

ECONOMIA DE ENERGIA E GRAVAÇÃO REMOTA DE DISPOSITIVOS ZIGBEE VISANDO A AUTOMAÇÃO PREDIAL

Lucas Guilhem de Matos
Ney César de Melo Filho

Brasília, Abril de 2013

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ECONOMIA DE ENERGIA E GRAVAÇÃO REMOTA DE DISPOSITIVOS ZIGBEE VISANDO A AUTOMAÇÃO PREDIAL

Lucas Guilhem de Matos
Ney César de Melo Filho

Relatório submetido como requisito parcial para obtenção
de grau de Engenheiro Mecatrônico

Banca Examinadora

Prof. Adolfo Bauchspiess (Orientador)

Prof. Fulano de Tal 2, UnB/ Dep

Prof. Fulano de Tal 3, UnB/ Dep

Dedicatória(s)

A inclusão de uma página de dedicatória é opcional e fica a critério de cada autor. No caso de optar-se pela inclusão da mesma, este espaço, nesta formatação, deve ser empregado para tal. Para trabalhos em dupla, o autor 1 deve preencher a coluna a direita enquanto o autor 2 preencherá a coluna a esquerda.

Ney César de Melo Filho

A inclusão de uma página de dedicatória é opcional e fica a critério de cada autor. No caso de optar-se pela inclusão da mesma, este espaço, nesta formatação, deve ser empregado para tal. Para trabalhos em dupla, o autor 1 deve preencher a coluna a direita enquanto o autor 2 preencherá a coluna a esquerda.

Lucas Guilhem de Matos

Aradecimentos

Agradeço a Blá-blá blá blá blá!!!

Lucas Guilhem de Matos

Eu também

Ney César de Melo Filho

RESUMO

A utilização de redes com tecnologia sem fio está cada vez mais difundida no mundo da automação. Este trabalho tem o intuito de mostrar como dispositivos sem fios podem ser utilizados em redes de automação predial, bem como mostrar quais são as vantagens oferecidas e os problemas encontrados.

O que se fez na primeira parte deste trabalho de graduação foi um estudo aprofundado das características energéticas de cada tipo de dispositivo de rede, analisando possíveis falhas de programação que possam prejudicar a confiabilidade na transmissão de dados e que possam aumentar o gasto de energia em cada dispositivo. Para isso, utilizaram-se módulos ZigBit TM, que são dispositivos sem fio que se comunicam por meio do protocolo ZigBee, programáveis através da pilha de bibliotecas BitCloud, ambos desenvolvidos pela Atmel. Este trabalho trata da aplicação do modo domir nestes módulos, mostrando resultados experimentais e devidas análises para diversas situações aplicáveis ao meio predial. Para isso, foram construídas placas para gravação de firmware e para comunicação serial com o computador e também placas que serviram como dispositivos remotos de rede, tais como roteadores e nós sensores da rede. Foram feitos experimentos modificando-se seus tempos de dormir e DuCy para saber qual é o real efeito de cada variável no dispositivo e na rede.

Na segunda parte do trabalho, foi aplicada a funcionalidade de atualização remota de firmware. Para tal, foram utilizados módulos ZigBit acoplados a kits de desenvolvimento MeshBean, também da Atmel. Foi feito experimento para descobrir o quão demorado pode ser a gravação sobre o ar e também para saber se todos os pacotes de dados enviados ao módulo a ser atualizado chegam ao seu destino por inteiro.

palavras-chave: ZigBee, redes sem fio, automação predial, economia de energia, gravação remota, ZigBit, BitCloud, modo dormir.

ABSTRACT

The use of networks with wireless technology is increasingly pervasive in the world of automation. This work aims to show how wireless devices can be used in building automation networks, as well as showing what are the advantages and problems.

What was done in the first part of this graduate work was a detailed study of the energy characteristics of each type of network device, analyzing possible programming flaws that might affect the reliability of data transmission and that may increase energy expenditure in each device. For this, we used ZigBit TM modules, which are wireless devices that communicate using the ZigBee protocol, programmable via BitCloud cluster libraries, both developed by Atmel. This paper addresses the implementation of sleep mode in each of these modules, showing experimental results and analysis for different situations applyable through building automation. For this, boards were built for recording and firmware for serial communication with the computer and also plates that served as remote network devices such as routers and sensor nodes of the network. Experiments were done by modifying their times of sleep and duty cycle to know what is the real effect of each variable on the device and the network.

In the second part, we applied the functionality of remote firmware update. To this end, we used ZigBit modules coupled to development kits MeshBean also from Atmel. Experiment was done to find out how much time it can consume to write over the air and also to know if all the packets of data sent to the module to be updated arrive at their destination in full.

keywords: ZigBee, wireless network, building automation, energy savings, remote upgrade, ZigBit, BitCloud, sleep mode.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

ADC	*Analog to Digital Converter*
API	*Application Programming Interface*
APS	*Application Support Sub-Layer*
ASHRAE	*American Society for Heating*
AVR	*Alf Vegard RISC processor*
BACnet	*Building and Automation Controls Network*
BAS	*Building Automation System*
BSP	*Board Support Package*
DuCy	*Duty Cycle*
EEPROM	*Electrically Erasable Programmable Read Only Memory*
GPI/O	*General Pin Input/Output*
HAL	*Hardware Abstraction Layer*
HVAC	*Heating, Ventilation and Air Conditioning*
I²C	*Inter Integrated Circuits*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
IRQ	*Interruption Query*
ISD	*Image Storage Driver*
JTAG	*Join Test Action Group*
LED	*Light Emitting Diode*
LPRF	*Low Power Rate Frequency*
OTAU	*Over-The-Air Upgrade*
P2P	*Peer to Peer*
PC	*Personal Computer*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computing*
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchronous Serial Transceiver*
USART	*Universal Synchronous/Asynchronous Serial Transceiver*
VoIP	*Voice over Internet Protocol*
Wi-Fi	*Wireless Fidelity*
WSN	*Wireless Sensor Network*
ZCL	*ZigBee Cluster Library*
ZDO	*ZigBee Device Object*

1 INTRODUÇÃO

Este capítulo apresenta considerações gerais sobre motivações que foram levadas em conta para a elaboração, bem como o objetivo geral do trabalho. São abordados assuntos como o estado em que se encontram os estudos na área

1.1 CONTEXTUALIZAÇÃO

A busca por conforto e comodidade é cada vez mais comum. É neste intuito que surge o conceito de automação, tanto residencial como predial. A busca por conforto térmico, segurança e praticidade no dia-a-dia tem aumentado a procura pela solução da automação. É, portanto, uma área cada vez mais difundida nos prédios e residências.

O conceito de automação não está ligado somente ao aumento da eficiência na produção de bens e produtos. Além disso, estão diretamente conectados conceitos como qualidade, conforto e economia. É importante destacar que a automação não está ligada somente à área industrial.

Muitos são os estudos na área de automação predial e de eficiência energética. Em geral, diz-se que em um sistema de automação predial (BAS), a máxima eficiência energética é atingida quando três pilares estão integrados, a Envoltória, o Sistema de Ar Condicionado e o Sistema de Iluminação, sendo possível monitorá-los e controla-los como se queira.

A envoltória diz respeito à construção do edifício. É possível de se reduzir de maneira significativa os gastos energéticos de um prédio a partir de sua construção. A utilização de vidraças, por exemplo, para que durante o dia não seja necessária a utilização de luminárias diminui os gastos com energia elétrica, bem como um projeto de ar condicionado onde tubulações de água gelada possuem material que dissipam pouco calor, para que *chillers* e torres de resfriamento possam trabalhar menos. Isso garante boa eficiência energética.

O sistema de ar condicionado de um prédio corresponde a parte considerável de gastos de energia. Mesmo depois da construção de todo esse sistema, é possível ainda melhorar a eficiência energética a partir de redes de monitoramento de seus equipamentos. Um bom sistema de ar condicionado possui, integrado a ele, um sistema de monitoramento e controle. Com ele torna-se possível encontrar padrões de tendências, para montar grades de agendamento capazes de controlar todo o prédio. Isso permite que se tenha baixo custo de operação e diminuição nos gastos de energia.

O sistema de iluminação também corresponde a gastos consideráveis de energia. Da mesma maneira, um bom sistema de iluminação é aquele que possui um sistema de monitoramento e controle acoplado a ele.

É claro que um sistema predial é bem mais complexo do que isso. Além destes pilares, outros sistemas são de suma importância: Sistema de Controle de Acesso, Sistema de Circuito Fechado de TV, Sistema de Detecção e Alarme de Incêndio e até mesmo Sistemas de Integração de Sistemas.

Figura 1.1. Sistemas de Automação Predial. [14]

O escopo deste trabalho abrange conceitos importantes e aplicáveis não só ao meio industrial, mas principalmente predial e residencial. Ao contrário do que se espera, nenhuma das vantagens de um ambiente inteligente depende de um meio físico, como máquinas e controladoras de última geração: hardware nem sempre gera benefícios, são as aplicações que geram. Passou-se, então, a ser percebida a possibilidade de se aplicar a automação para controlar e monitorar o desempenho de um prédio.

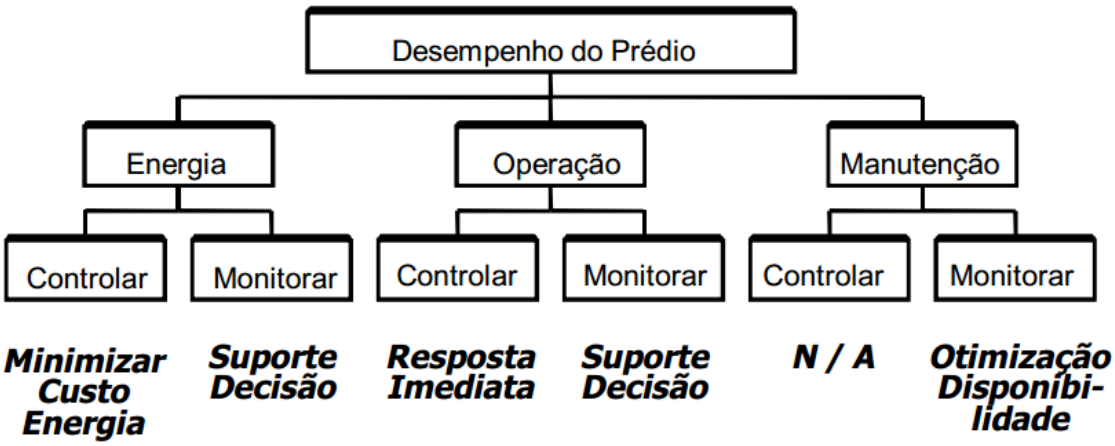


Figura 1.2. Aplicações para melhoria do desempenho predial. [11]

É possível perceber que o homem está cada vez mais preocupado com o meio ambiente. Surge, então, o conceito de sustentabilidade aplicada a qualquer área da ciência. Uma das maiores dificuldades na implementação de um sistema automatizado se encontra na busca pela economia de energia. Um dos grandes motivos pela busca da automação em ambientes industriais e prediais é a economia de energia.

Existem muitas linhas de pesquisa sobre monitoramento predial. Todas elas levam em conta que a Operação é o ponto chave desse monitoramento. Logo no início de vida de um prédio, os custos com operação são mínimos se comparados com os custos de construção do mesmo. Mas se for levado em conta um ciclo de vida de um prédio de 40 anos, é possível ver onde estão os maiores custos.

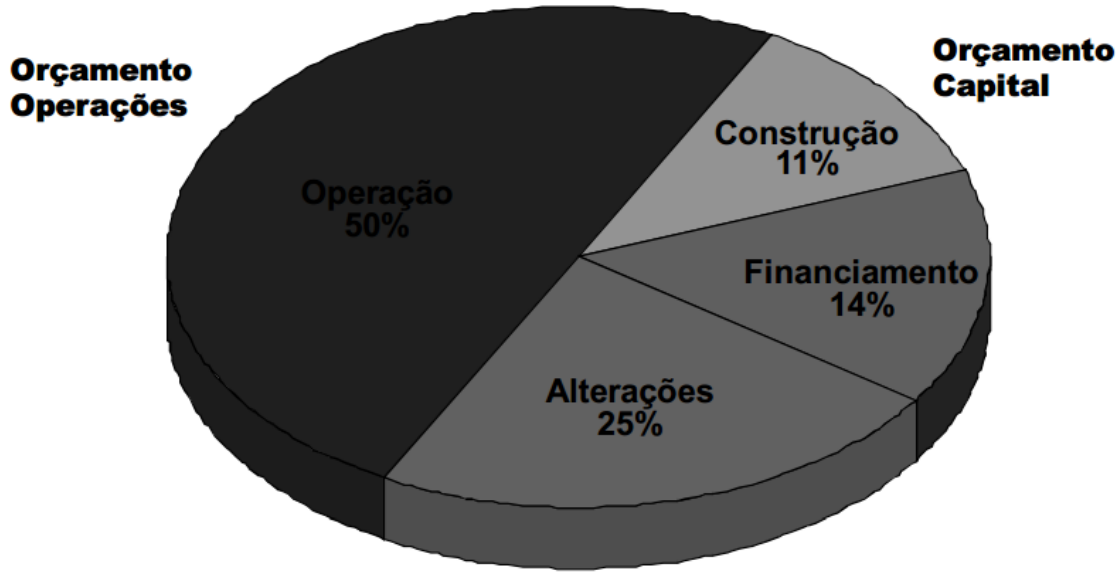


Figura 1.3. Porcentagem de custos acumulados de um prédio típico de 40 anos. [11]

A Sociedade Americana de Engenharia para Aquecimento, Refrigeração e Ar Condicionado (ASHARE) destaca ainda que os custos com iluminação e climatização são os responsáveis por 64% dos custos totais de operação. Dentre os custos totalizados em um prédio de 40 anos, ao menos 35% é relativo ao uso de energia elétrica. [11]

Ou seja, o desafio da automação predial é reduzir os custos de operação. Isto é feito a partir de controle de demanda, programação horária, revezamento de equipamentos, relatórios específicos de análise de energia e também pelo controle energético das redes de automação.

Outros grandes problemas relacionados à automação predial são diagnóstico e manutenção de equipamentos de rede, como controladoras de campo, ou até mesmo problemas estruturais de rede, como cabos rompidos ou falha de comunicação entre dispositivos.

Em geral, resolve-se este tipo de problema com operação em campo, tentando encontrar manualmente a razão do problema para então atuar em cima disso e consertar a rede.

A capacidade de se diagnosticar e solucionar o problema de rede dentro de uma estação de trabalho remota é algo que já existia antes da invenção da automação sem fio. Portanto, deve ser necessária a possibilidade de se programar ou de se atualizar um nó (dispositivo) danificado da rede mesmo em redes wireless.

Isso pode ser feito com o auxílio de rotinas de monitoramento oferecidas por softwares específicos e também a partir da correta programação de cada nó, tornando-o mais flexível a possíveis mudanças de comportamento e a resolução de problemas de modo imediato.

Dentro do contexto visto e explicado, o objetivo geral deste trabalho de graduação é o enfoque em maneiras de aplicar a tecnologia sem fio nos processos de automação predial e residencial, visando diminuir o gasto de energia nos dispositivos envolvidos na rede de automação e implementando a funcionalidade de programação remota em cada um deles.

1.2 ESTADO DA ARTE

Os estudos, bem como a tecnologia dos dispositivos ZigBee, são bastante recentes. A ideia de rede ZigBee começou a surgir em 1998, com a necessidade de aplicação wireless no segmento da automação. Percebeu-se, naquele momento que protocolos já existentes, tais como Bluetooth e Wi-Fi não eram suficientes, ou ao menos aplicáveis. [12]

Mas foi somente em 2003, com o surgimento da IEEE 802.15.4, que se passou a investir na ideia. Durante a última década, dispositivos ZigBee vêm sendo utilizados principalmente em aplicações residenciais.

O que se espera para o futuro da automação predial é:

- Redes wireless facilmente expansíveis;
- Pouca necessidade de construção de infraestrutura com cabeamento;
- Baixo custo de implementação;
- Possibilidade de aplicação em praticamente qualquer ambiente ou sistema de automação;
- Baixo consumo de energia;
- Retorno financeiro, que seria a relação entre o investimento feito para automatizar o sistema e a economia energética obtida com o tempo, ainda mais rápido.

O padrão ZigBee de comunicação está em crescente expansão, sendo cada vez mais utilizado para diferentes aplicações. Isto não só porque atende aos fatores citado acima, mas também devido a sua praticidade e bom padrão de segurança da rede. Além disso, possui um padrão de comunicação com foco direto em monitoramento e controle de redes e, principalmente, em aplicações para sensores.

Outra qualidade da rede é a padronização em seu protocolo, no que diz respeito às camadas de rede e de aplicação. Em outras palavras, placas de diferentes modelos e até mesmo de diferentes fabricantes são capazes de se encontrar na rede automaticamente, se comunicando sem algum problema. É o que a ZigBee Aliance chama de ZigBee Standard (em português, padronização).

A ZigBee Aliance é o grupo que cuida da normalização e cria os modelos a serem seguidos pelos fabricantes. É também o grupo responsável pela maioria das pesquisas no segmento de automação wireless no mundo. Tal modelamento dos dispositivos e do protocolo de comunicação, segundo o grupo, é o responsável pelo crescimento das redes wireless ao redor do globo, bem como pela vasta gama na variedade e na inovação dos produtos e dispositivos ZigBee. Grandes estudos envolvendo a tecnologia wireless aplicada à automação predial estão em desenvolvimento, mas poucos concluídos.

Alguns dos estudos na área de edifícios comerciais foram concretizados ainda em 2011, por pesquisadores participantes do projeto da ZigBee Alliance. Porém, ainda existem muitas

barreiras a serem transpostas no segmento. Por exemplo, todos os estudos feitos na área são baseados em outros protocolos, principalmente no que se trata de sistemas supervisórios de monitoramento e integração de diferentes áreas, como segurança predial (incêndio, controle de acesso e sistema de câmeras e TV) e sistema de Ar Condicionado (chillers, fancoils, etc). O protocolo ZigBee fica em um plano secundário; o sistema de controle tem ainda como base outros protocolos de comunicação, como BACnet, LonWorks, N2 e até Modbus, em redes mais antigas.

Ainda assim, muitas das grandes empresas de automação predial mundiais estão investindo na ideia. Estudos na área de redes wireless em automação predial na tentativa de se atingir um ambiente inteligente ainda não estão completamente desenvolvidos, mas os esforços estão cada vez maiores. Empresas como Honeywell, Ingersoll-Rand/Trane, Johnson Controls, Schneider Electric e Siemens (líderes mundiais no segmento) dão suporte no desenvolvimento de novos produtos e são parceiras da ZigBee Alliance. A Honeywell é uma das marcas que mais vende painéis e detectores de incêndio no mundo. Trane é das empresas mais conceituadas no que se diz respeito à produção de Chillers. Johnson Controls é líder na área de eficiência predial. Ou seja, grandes empresas com negócios afins, mas aplicações diferentes, investindo neste novo desafio. Isso diz muito a respeito do que está por vir em termos de novos estudos, novas tecnologias e novas aplicações de redes wireless. [15]

Alguns estudos já foram feitos, também na Universidade de Brasília. No Trabalho de Graduação dos alunos José Urbano Duarte Junior e Mariana Carolina Carvalho Novais, com título Instrumentação e Controle de um Sistema de Ar Condicionado Híbrido utilizando BACnet sobre ZigBee, algumas soluções de redes wireless aplicadas à automação predial foram utilizadas. O trabalho tem como foco o conforto térmico proporcionado por automação baseado em dispositivos sem fio. Assim como já foi citado anteriormente, é um estudo que coloca o protocolo ZigBee em segundo plano. A rede principal de monitoramento e controle é o BACnet. [4]

Ainda neste trabalho, os autores ressaltaram que o modelo de comunicação de BACnet sobre ZigBee é apenas satisfatório. Disseram que são redes com propósitos distintos, onde BACnet se baseia na segurança e aplicabilidade da rede, enquanto que a outra enfatiza a economia de energia de seus dispositivos. Como proposta para trabalhos futuros, o trabalho expõe o ponto de vista dos autores de que o modo dormir, utilizando timeouts ainda maiores, deixaria a comunicação ainda pior. Ou seja, a utilização do modelo BACnet sobre ZigBee não é ideal, uma vez que a busca pela economia máxima de energia por dispositivo atrapalharia a comunicação entre os dois protocolos. [4]

Outro Trabalho de Graduação bastante importante no sentido de documentação e aplicação do modelo de redes ZigBee em edifícios é o trabalho do aluno Davi Stoll Evangelista, intitulado Integração de Redes de Sensores ZigBee para Automação Predial Utilizando Módulos Meshbean. Este tem como problema principal a comunicação entre redes ZigBee distintas, de modo que possam trabalhar conjuntamente em ambientes confortáveis. Assim como no outro trabalho, o foco está também no controle de um ar condicionado e no conforto térmico gerado. [3]

É mostrada, pelo autor, a dificuldade de comunicação entre as redes, que devem se revezar entre a medição de dados e emparelhamento com a rede vizinha, e também a dificuldade encontrada no processamento dos dados para posterior controle. É um estudo bastante importante na área, uma vez que enfatiza a descentralização dos módulos de controle de malhas de redes, algo necessário para dispositivos de baixo alcance. [3]

Algo importante a ser citado é a compatibilidade com modo dormir (para economia de energia) de dispositivos roteadores e finais. Drew Gislason, grande estudioso na área de

protocolos de comunicação sem fio, mostra em seus estudos [1] que os nós das redes podem estar dormindo durante alguns intervalos de tempo que de nada atrapalha a intercomunicação das redes. Em uma rede com tratamento de dados, a confiabilidade é ainda maior, mesmo em modo dormir.

2 ESPECIFICAÇÃO DE PROJETO E APRESENTAÇÃO DO ZigBit™

Neste módulo, serão apresentados os motivos da escolha dos módulos ZigBit como base de pesquisa durante o decorrer do trabalho.

O objetivo deste trabalho é a criação de uma rede básica de automação predial que contenha uma série de caraterísticas a serem explanadas adiante. O motivo para a escolha de tais características é a economia e facilidade de uso da rede, assim como a robustez da troca de dados.

A primeira característica procurada é o baixo consumo de energia dos dispositivos que se encontram na rede de modo que os dispositivos que funcionam como Dispositivo-Final tenham um funcionamento, utilizando-se uma bateria alcalina de 9V, contínuo sem troca de bateria por um período próximo ao de um ano. Para tal, utilizando uma bateria de 9V alcalina com aproximadamente 500mAh de carga, teríamos que ter no máximo um consumo médio de 57uA no dispositivo.

A segunda característica esperada da rede é que os dispositivos possuam a funcionalidade de gravação remota. É importante fator de escolha que os dispositivos possam funcionar dessa maneira para a maior facilidade de manipulação e utilização da rede.

Além das funcionalidades antes descritas, necessita-se de uma rede que seja flexível em relação à adição de novos nós e dispositivos, e tenha uma alta segurança de troca de informações, ou seja, poucos pacotes perdidos.

A solução de automação sem fio tem se mostrado de bom custo benefício para sistemas de controle, dando mais mobilidade à rede principal de automação e trazendo interoperabilidade aos dispositivos para se comunicarem na mesma rede.

Acredita-se que, assim como redes wireless já dominam o mercado de equipamentos de acesso à internet com sua praticidade, as redes de automação sem fio também podem. Seu potencial está basicamente nas mesmas características: facilidade de interconexão, possibilidade de comunicação em qualquer lugar dentro do alcance da rede sem a necessidade de construção de infraestrutura ou cabeamento, confiabilidade na transmissão e recebimento dos pacotes de dados e baixo custo.

Além de tudo isso, a aplicação wireless dá vantagens em sistemas com características específicas, como: sistemas com interconexão de múltiplos edifícios, ambientes onde a infraestrutura é de difícil acesso, ou fica caro, e até em casos em que existe necessidade de mobilidade. Em geral, o sistema de gerenciamento de um prédio não é feito somente em um tipo de rede. Para o controle e a automação, utiliza-se uma rede voltada para o problema, tais como LON e BACnet, mas sistemas de câmera (para monitoramento por TV) ou tarefas de uso geral são tipo VoIP (*Voice over IP*). A utilização de uma rede sem fio que capaz de fornecer largura de banda suficiente com boa qualidade possibilita a implementação de uma única rede para controle, dados e voz.

É tendo essa ideia em vista que o princípio adotado para a implementação do trabalho foi a utilização de rede wireless, e não qualquer rede wireless, mas uma rede wireless que tenha ênfase em baixo consumo de energia e que também tenha uma plataforma de aplicação extensa de modo que fosse possível a flexibilidade na programação da rede e de modo que se permitisse o processo de gravação remota.

Dentre as opções de protocolos de rede wireless de baixo consumo de energia, optou-se por usar o protocolo ZigBee, não somente pelo baixo consumo, mas também pelo modo como a rede funciona. Os nós da rede são completamente independentes, se um nó se perde da rede os outros continuam funcionando normalmente, o que atende a priori nossas necessidades.

A maior vantagem ao se utilizar uma rede ZigBee é a capacidade de expansão da rede. É possível de se adicionar quantos dispositivos se queira à rede, de modo a aumentar o alcance total da mesma ou a aumentar o número de equipamentos monitorados. Cada dispositivo entra com um novo endereço na rede de forma automática, sem necessidade de comissionamento ou algo parecido, desde que as placas ZigBee sejam compatíveis e estejam programadas para se conectar.

Outra grande vantagem de se usar a rede ZigBee é que seu protocolo de comunicação é bem robusto, a confiabilidade da transmissão é muito alta. A rede continuará funcionando e trocando informações com altos níveis de sucesso mesmo ambientes que possuam condições adversas como um laboratório onde se encontram inúmeras frequências de rádio "poluindo" o ambiente, ou ainda mais importante, essas condições são ainda mais fundamentais quando se trata de um ambiente comercial ou industrial.

A rede ZigBee tem como consequência de sua característica independente, a possibilidade de trabalhar me diversas topologias, por exemplo, estrela, cluster, etc.

Dentre estas vantagens a que mais chama atenção é a de baixo consumo de energia. A rede ZigBee tem como filosofia o baixo consumo. Embora o consumo de energia varie de dispositivo para dispositivo, mesmo o dispositivo ZigBee que mais utiliza carga consome muito pouco. Um bom projeto de hardware para estes dispositivos possibilita um gasto de corrente mínimo.

É imprescindível também comentar que os dispositivos ZigBee são comumente dispositivos de baixo custo, e a implementação da rede também.

Existem claro as desvantagens. As redes wireless normalmente tem uma capacidade de transmissão de dados menor do que as redes cabeadas. No caso do ZigBee a capacidade de fluxo de dados é muito menor mesmo do que algumas redes wireless, no entanto, no escopo da nossa aplicação e na possibilidade de fazermos um controle distribuído de maneira a reduzir a troca de mensagens constante, a rede ZigBee, mesmo com baixa capacidade de fluxo de dados é a opção mais chamativa.

Outra desvantagem da rede ZigBee é que esta possui um alcance reduzido em comparação com outros protocolos de rede wireless. No escopo da aplicação abordada nesse documento o alcance não é um empecilho para o trabalho, ou seja, as desvantagens que a rede ZigBee apresenta não se aplicam diretamente ao ambiente de trabalho escolhido, mas suas vantagens sim, e em face dessas vantagens que condizem com as exigências do trabalho, escolhemos a rede ZigBee.

O próximo passo para o início do trabalho é a escolha de um dispositivo que funcione com o protocolo ZigBee.

Usando como prioridade a necessidade de uma plataforma de desenvolvimento com grande gama de aplicação e suporte, encontramos o dispositivo ZigBit. O ZigBit é produzido pela Atmel e possui a plataforma de desenvolvimento chamada BitCloud. A BitCloud nada mais é do que um software embarcado que permite o desenvolvimento de firmwares para aplicações wireless. É baseado na programação em C, ou seja, é uma pilha de bibliotecas estruturadas e direcionadas à funcionalidade e eficiência das aplicações. A característica "*event-driven*" do

ZigBit é, na verdade, uma herança do tipo de programação embutida a ele, baseada em linguagem C.

As vantagens encontradas em se utilizar o ZigBit como dispositivo escolhido começam em seu custo. O ZigBit tem baixo custo, principalmente se visualizarmos a funcionalidade e capacidade do dispositivo, com um ATMega1281 embarcado e com uma plataforma de desenvolvimento extensa.

As principais vantagens do ZigBit são: baixíssimo consumo de energia, chegando consumir 6uA de corrente em seu estado inativo, altíssima sensibilidade de recepção o que ajuda no índice de sucesso de transmissão, uma característica reforçada do protocolo ZigBee, recursos de memória extensos e pinos auxiliares para as mais variadas aplicações como GPI/O, SPI, USART/UART I²C entre outros e finalmente a antena que possui ganho baixíssimo mas mesmo assim tem um alcance considerável.

Possui, também, a capacidade de ser programado com funcionalidade atualização remota, o que se chama de Over-The-Air Upgrade (OTAU), ou seja, a capacidade de cada nó da rede ser alterado da maneira como se deseja sem a necessidade de estar ligado por meio físico a um computador, responsável pela atualização de cada módulo.

A desvantagens em se trabalhar com o ZigBit deriva de sua melhor vantagem. Ao utilizar a BitCloud, que é uma pilha de bibliotecas utilizada como suporte para a programação de cada módulo (melhor explicado mais à frente), mesmo com uma gama extensa de aplicações e documentação, a programação do ZigBit se torna uma "caixa preta" para os desenvolvedores. As funções são facilmente compreendidas em altíssimo nível, mas não se sabe exatamente qual o programa por trás delas e isso dificulta o processo de *debug* e tratamento de falhas.

Ainda em relação à maneira de se programar, o desenvolvedor está restrito a utilizar a lógica de funcionamento da BitCloud. Como descrito anteriormente, a BitCloud mesmo com sua limitação permite uma gama extensa de aplicações, que supri praticamente todas as aplicações que estão no escopo de estudo deste documento, mas ainda assim, acaba por se tornar pouco maleável em situações em que o desenvolvedor pretende fazer algo muito específico ou trabalhar de maneira "não ortodoxa" em relação à maneira como as aplicações da Atmel trabalham.

Em vista destas vantagens e desvantagens verifica-se que a rede ZigBee tendo como representante o dispositivo ZigBit supre, a priori, as necessidades primárias determinadas no início deste projeto, e por isso, foi a escolhida para o desenvolvimento que se segue.

3 ZigBee UTILIZANDO ZigBit™

Neste capítulo, são citados alguns dos conceitos fundamentais sobre redes ZigBee e também sobre os dispositivos ZigBit. Alguns dos termos citados aqui serão utilizados ao decorrer de todo o restante do relatório. Serão explicadas funcionalidades, aplicações e modos de programação dos módulos.

3.1 A REDE ZigBee

As redes ZigBee aparecem em um novo nível no que se trata sobre economia de energia. Dispositivos ZigBee são de baixa potência de funcionamento e, por isso, são perfeitos para a aplicação que se deseja: automação de um ambiente de maneira a deixa-lo mais confortável e ainda mais econômico.

ZigBee é nada mais do que um protocolo de comunicação em rede de dispositivos de baixo consumo de energia que se comunicam através de ondas eletromagnéticas, ou seja, wireless (sem fio).

O protocolo ZigBee é baseado no padrão IEEE 802.15.4, que designa protocolos de transferência de dados de camadas mais inferiores, como camadas de acesso físico e de acesso médio de controle de dados.

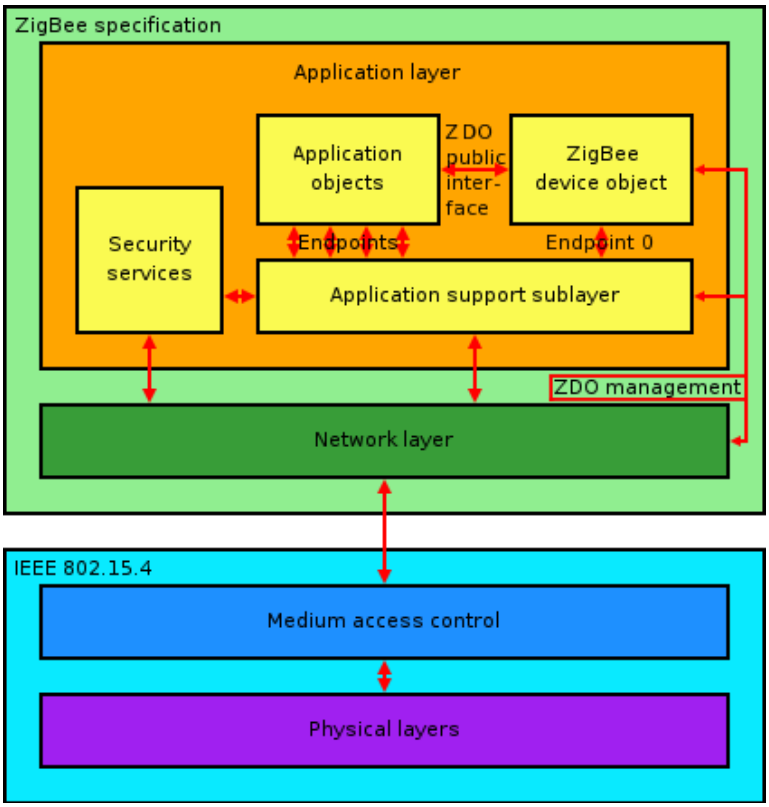


Figura 3.1. Padrão IEEE 802.15.4 aplicado a redes ZigBee. [16]

O padrão IEEE é responsável pelo acesso físico de dados e também pelo controle da transferência (recebimento e envio de pacotes), enquanto que a especificação ZigBee é a responsável pela camada de rede (acesso e controle de diferentes dispositivos na rede) e pela camada de aplicação, onde se preocupa com a interface com o usuário, com a segurança (criptografia) da rede e com suporte.

Redes ZigBee se encaixam na definição LPRF (*Low Power Rate Frequency*), ou seja, é de baixo consumo de energia e com baixa frequência de transmissão de dados. Baixa taxa de transmissão significa que a comunicação é lenta. Por ser de baixa potência, o alcance também não é grande. Mas nenhum desses quesitos atrapalha a implementação de uma rede de automação.

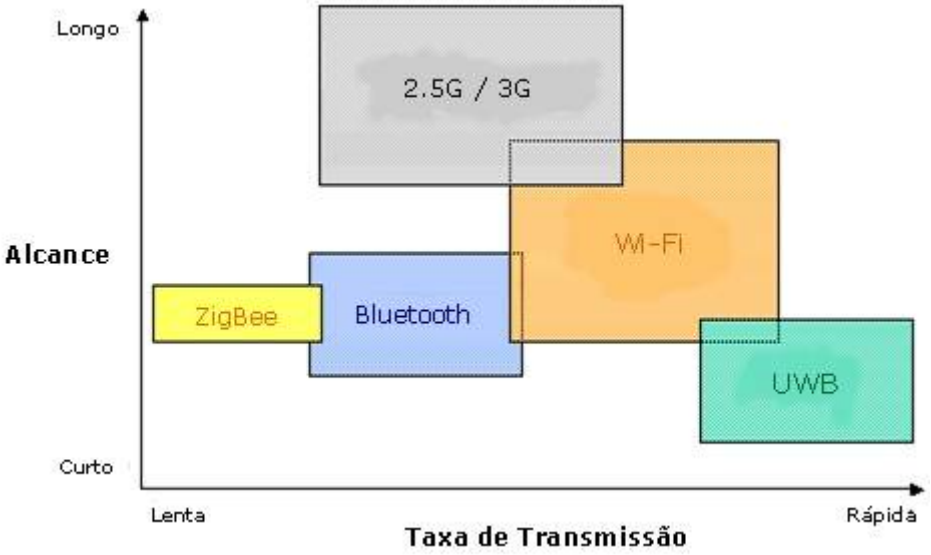


Figura 3.2. Taxa de Transmissão X Alcance. [17]

Dispositivos ZigBee possuem alcance baixo, normalmente de 30 a 200 metros em ambientes fechados, dependendo do tipo de dispositivo. Em ambientes abertos, pode-se chegar a cerca de 1,5 Km de alcance. De qualquer forma, não é um alcance grande, mas é possível adicionar quantos roteadores quanto se queira para expandir a rede, podendo ter aplicações em ambientes residenciais, prediais e até mesmo agrícola.

Existem basicamente três tipos de dispositivos ZigBee: coordenador, roteador e dispositivo final. Coordenadores são responsáveis por controlar a transmissão de dados entre dispositivos e também por receber e repassar os dados que vêm dos nós das pontas (dispositivos finais). Os roteadores são responsáveis apenas por repassar dados recebidos pelo nó anterior para o próximo nó da rede.

Os dispositivos finais (*end devices*) são os nós da ponta da rede. Neles são medidos dados importantes para a rede de automação, tais como temperatura ou presença de fumaça, por exemplo. Portanto, são neles que ficam posicionados sensores de medição. É importante destacar que esses dispositivos não são responsáveis por qualquer processamento de dados, mas somente pela medição e transmissão do dado medido. Além disso, nos dispositivos finais se encontram os atuadores, que controlam o comportamento de diversos tipos de equipamentos dentro de um sistema.

Seguindo esta tendência, tem-se ainda tipos de redes, separadas por diferentes topologias. Existem 3 topologias normalmente utilizadas: topologia em árvore, em estrela e em malha.

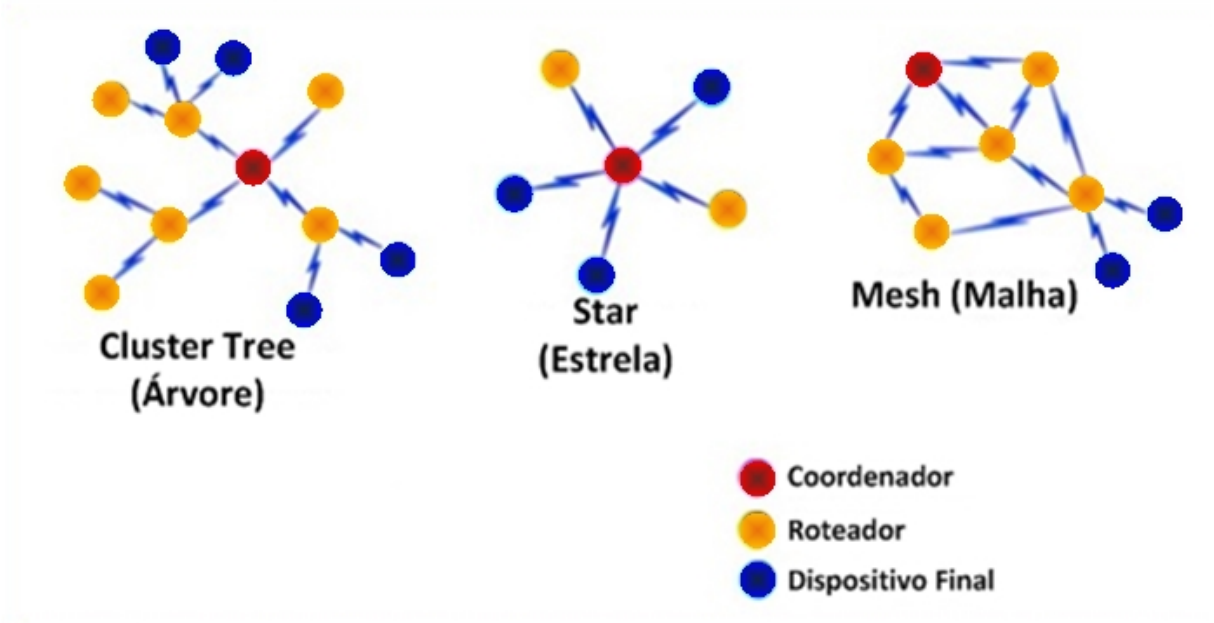


Figura 3.3. Topologia de Redes ZigBee. [18]

A topologia em árvore á a mais utilizada, pois é de grande alcance (pode-se adicionar à rede quantos roteadores se queira) e de fácil implementação. Em estrela é muito utilizada para ambientes fechados e pequenos, pois o coordenador está ligado somente a dispositivos considerados finais. Em malha é a topologia mais complexa, porém mais eficiente, pois possibilita um bom funcionamento da rede mesmo se alguns dispositivos intermediários estejam com defeito.

Módulos ZigBee são programáveis, o que possibilita aplicações em uma gama muito grande de diferentes sistemas. Sua programação é do tipo *event-driven*, ou seja, baseia-se no acontecimento de certos eventos, sendo eles que ditam o comportamento da rede e do processamento de dados. Desta forma, o fim da execução de uma tarefa leva ao início de uma nova tarefa subsequente de maneira linear.

3.2 BitCloud

A programação de cada módulo é feita separadamente de acordo com a aplicação que se queira utilizar, contando ainda com o auxílio de pilhas de bibliotecas (APIs), fornecidas como modelo de estudo. As pilhas de bibliotecas mais conhecidas são a Z-stack e o BitCloud. Cada fornecedor possui sua própria API, mas todas elas são interoperáveis. Isso significa que caso se utilize uma API do fornecedor *Texas Instruments* em um equipamento Atmel, deverá funcionar.

O BitCloud é uma pilha de software embarcada, completa em termos de aplicação. A pilha promove desenvolvimento de firmwares, para aplicações com segurança de dados, confiabilidade de rede e escalonamento de tarefas, que é baixada para hardwares da Atmel, como o ZigBit.

Isso quer dizer que qualquer tipo de segurança de rede é feito a nível de software e toda a criptografia é feita a partir da pilha BitCloud. Assim como a confiabilidade de rede, que é a capacidade da rede de garantir que qualquer pacote de dados processados cheguem até seu destino.

Um recurso bastante importante presente na BitCloud é o escalonamento: se existe uma tarefa com maior prioridade de execução, existe um reajuste, causado pela concorrência de duas ou mais tarefas, seguida de interrupção e posterior chamada de uma nova tarefa, gerenciado pela pilha. Isso garante que todos os deadlines das tarefas sejam satisfeitas. Isso é o que se chama de *Kernel Multi-Task* simulado. Ou seja, apesar de não possuir hardware multiprocessado para atender a diferentes tipos de tarefas ao mesmo tempo, existe a simulação em tempo real feito a partir do escalonamento de tarefas, o que deixa qualquer tipo de processamento de rotinas muito mais rápido, confiável e seguro, garantindo todos os tempos finais de execução.

É também a partir da BitCloud que é possível ajustar o hardware em questão à aplicação que se deseja. Neste caso, é a partir desta pilha de bibliotecas que é possível de se programar os dispositivos de rede para consumirem o mínimo possível de energia ou para embarcarem a capacidade de se atualizar o firmware remotamente.

A BitCloud corresponde à raiz da pilha, mas é na verdade um conjunto de diversos tipos de aplicações. Alguns de seus componentes são:

- APS: biblioteca que dá suporte à aplicação de sub-camadas a nível de protocolos de comunicação;
- BSP: pacote de suporte à placa MeshBean;
- ConfigServer: armazena parâmetros genéricos de configuração de dispositivos e, também, em nível de sub-camadas de rede;
- HAL (*Hardware Abstraction Layer*): dá suporte à camada física de acesso;
- MAC_PHY: suporta a camada de acesso médio e também camadas físicas;
- NWK: camada de rede;
- PersistDataServer: possibilita acesso à EEPROM e gerencia parâmetros de memória.
- Security: proporciona a segurança da rede, para que outros dispositivos não tenham acesso aos dados da rede;
- SystemEnvironment: diz respeito à funcionalidade principal de cada dispositivo, bem como é o gerenciador de tarefas;
- ZDO (*ZigBee Device Object*): proporciona iterface entre aplicação e a camada de rede;
- WSNDemo: aplicação WSN de aquisição de dados, segurança e gerenciamento de energia, funcionando em sinergia com o programa WSNMonitor para visualização de diversas funcionalidades;
- ZCL (ZigBee Cluster Library): framework presente em aplicações que envolvem agrupamento de tarefas usualmente importantes para diversos tipos de aplicação.

A plataforma da pilha de programação utilizada neste trabalho é a ATZB-DK-24 (ZDK), com a BitCloud 1.1.13

3.3 ZigBit™

O módulo utilizado é o ZigBit™ Amp OEM (ZDM-A1281-PN/PN0).

O ZigBit é um módulo compacto, de alcance estendido, baixo consumo e alta sensibilidade, com frequência de 2.4GHz de processamento e com protocolo de comunicação ZigBee (IEEE 802.15.4)

Pode ser, e já é utilizado, em diversas aplicações, como automação e monitoramento predial, monitoramento e controle HVAC e segurança, por exemplo.

Possui uma gama enorme de interfaces, como entradas GPIO (digitais), linhas IRQ (para interrupção), linhas ADC (conversores digital analógico), JTAG, porta UART com controle de dados, USART, linha I2C e SPI.

A comunicação via UART é um formato padrão para comunicação de dados de forma serial. São utilizados dois caminhos para transmissão de dados um em cada direção, em regime full-duplex. Os dispositivos que se comunicam entre si devem possuir mesma velocidade de transmissão e processamento de dados para que os pacotes de dados mantenham sua confiabilidade de recebimento e para que eles possam ser interpretados de maneira correta.

No modo USART, ocorre a transferência síncrona. Isso pode ser feito a partir de um caminho de dados, mas em regime half-duplex, ou seja, quando transmitindo, não há recepção de dados por parte do dispositivo.

O eZeeNet é seu software embarcado, com capacidade de programação remota (*Over-the-Air Upgrade*) e comandos AT (utilização de portas seriais para interface com o computador).

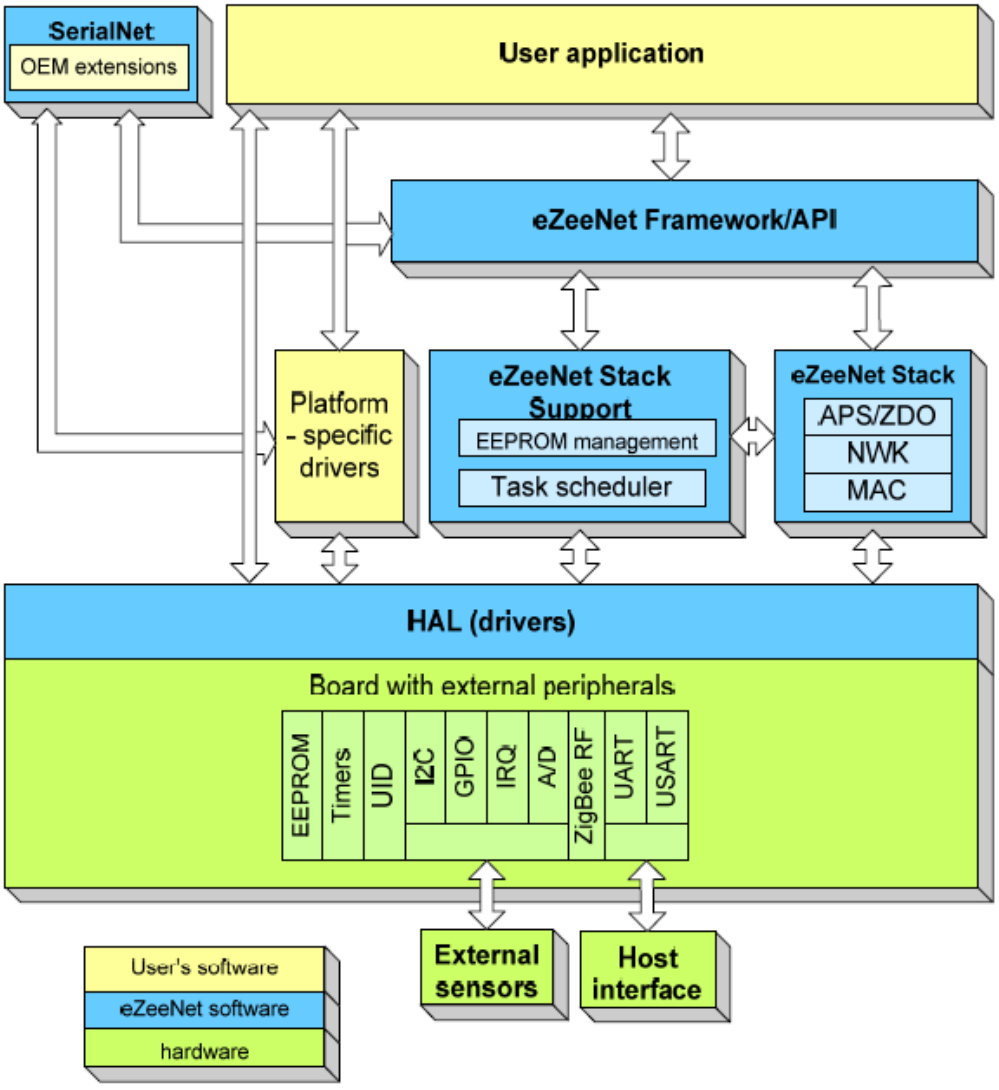


Figura 3.4. Esquemático do ZigBit. [7]

Seu alcance é de até 4 quilômetros em ambientes abertos, com flexibilidade para utilização de diferentes tipos de antena.

O ZigBit contém um módulo ATmega1281, que é um micro controlador AVR programável, bem como um transceptor (módulo de recebimento e envio de pacotes de dados) AT86RF230 com memórias Flash de 128K e RAM de 8K.

Algumas das especificações do módulo ZigBit são:

Parâmetros	Faixa	Unidade
Tensão de Alimentaçã	3.0 a 3.6	V
Consumo de Corrente: modo Rx	23	mA
Consumo de Corrente: modo Tx	50	mA
Consumo de Corrente: modo Dormir	<10	uA

Tabela 3.1 Parâmetros gerais do ZigBit.

Parâmetros	Faixa	Unidade	Condição
Banda de Frequência	2.400 a 2.486	GHz	
Número de Canais	16	Mhz	
Espaçamento de Canais	5	MHz	
Sensibilidade do Receptor**	-104	dBm	PER = 1%
Potência de saída do Transmissor	0 a +20	dBm	Ajustado em 16 passos
Taxa de dados "On-Air"	250	Kbps	
Impedância Nominal de TX Output/RX input	50	Ohms	Saída RF Desbalanceada
Alcance OutDoor	Até 4000	m	Com antena externa de 4.1 dBi

Tabela 3.2 Características do transceptor.

Parâmetros	Valor	Unidade
Tamanho da Memória Flash do Chip	128	Kbytes
Tamanho da RAM do chip	8	Kbytes
Tamanho da EEPROM do chip	4	Kbytes
Frequência de Operação	4	Mhz

Tabela 3.3 Características do microcontrolador.

Parâmetros	Valor	Unidade	Condição
Máximo Baud Rate da porta serial	38.4	Kbps	
Resolução/Tempo de Conversão do ADC	10/200	Bits/us	No modo de conversão única
Resistência de entrada do ADC	1	MOhm	
Tensão de referência do ADC	1.0 até Vcc -0.3	V	
Tensão de entrada do ADC	0 até Vref	V	
Clock Máximo do I²C	222	KHz	
Tensão de Saída dos GPIO	2.3/0.5	V	(-10/5 mA)
Frequência do Oscilador de Tempo Real	32.768	KHz	

Tabela 3.4 Características do módulo de interface.

Parâmetros	Valor	Nota
Tamanho, mm	38.0 x 13.5 x 2.0	
Peso, g		ZDM-A1281-PN
	~2	ZDM-A1281-PN0
Temperatura de Operação, ºC	-20 até 70	-40 até +85 é operacional
Faixa de humidade na qual opera	**Não mais que 80%**	

Tabela 3.5 Características Físicas do ZigBit.

4 MODO SLEEP DO ZigBit™

No capítulo 4 deste relatório, serão abordados descrição do problema, implementação, dados e resultados, assim como análise dos dados coletados para cada um dos experimentos feitos a respeito do sleep mode sobre módulos ZigBit.

4.1 O GASTO DE ENERGIA DO ZigBit

Um dos problemas que serão abordados neste trabalho é o gasto de energia do dispositivo ZigBit. Apesar de o ZigBit, assim como todos os dispositivos que funcionam com o protocolo ZigBee, ser um dispositivo que naturalmente consome pouca energia, em face dos motivos pelos quais escolhemos o dispositivo, sendo esses, o baixíssimo consumo de energia, a maleabilidade e principalmente a praticidade de trabalho, o seu consumo é, mesmo assim, muito maior do que gostaríamos que fosse.

O Problema do gasto de energia do ZigBit reside no fato de trabalharmos com redes maleáveis. Entenda-se por maleável uma rede em que os sensores pode ser movimentados e realocados com pouco esforço exigido e sem complicações para o bom andamento do trabalho. Para isso fazermos uso de alimentação por baterias ao invés de conectarmos o dispositivo em uma tomada, o que o deixaria preso aos arredores da mesma ou utilizaria um grande fio que seria um empecilho em qualquer troca de posição do sensor.

Com a ideia exposta anteriormente vem o questionamento a respeito do tempo de vida útil da bateria e de como potencializar este tempo. Primeiramente por causa do custo da bateria e também por causa da falta de praticidade que experimentamos ao trocar baterias diariamente.

Devemos então encontrar um meio de reduzir o consumo de energia do dispositivo drasticamente para que a troca de baterias não seja uma preocupação constante e nem o gasto com baterias seja muito elevado.

4.2 IMPLEMENTAÇÃO

4.2.1 HARDWARE UTILIZADO

Para trabalhar com uma rede devemos inicialmente, obviamente, cria-la. Para tal fim, determinamos as características de nossa rede e o que precisaríamos de cada dispositivo incluído nela e desenvolvemos a placa ZigBit Channel 1.0, uma placa de função genérica que será utilizada tanto para os módulos Coordenadores quanto para os módulos Dispositivo-Final. O módulo Coordenador também funciona como o módulo Gravador, onde gravamos os programas em cada um dos dispositivos. Uma imagem desta placa pode ser vista abaixo na Figura 4.1.

Figura 4.1. Placa em sua funcionalidade de Coordenador/Gravadora.

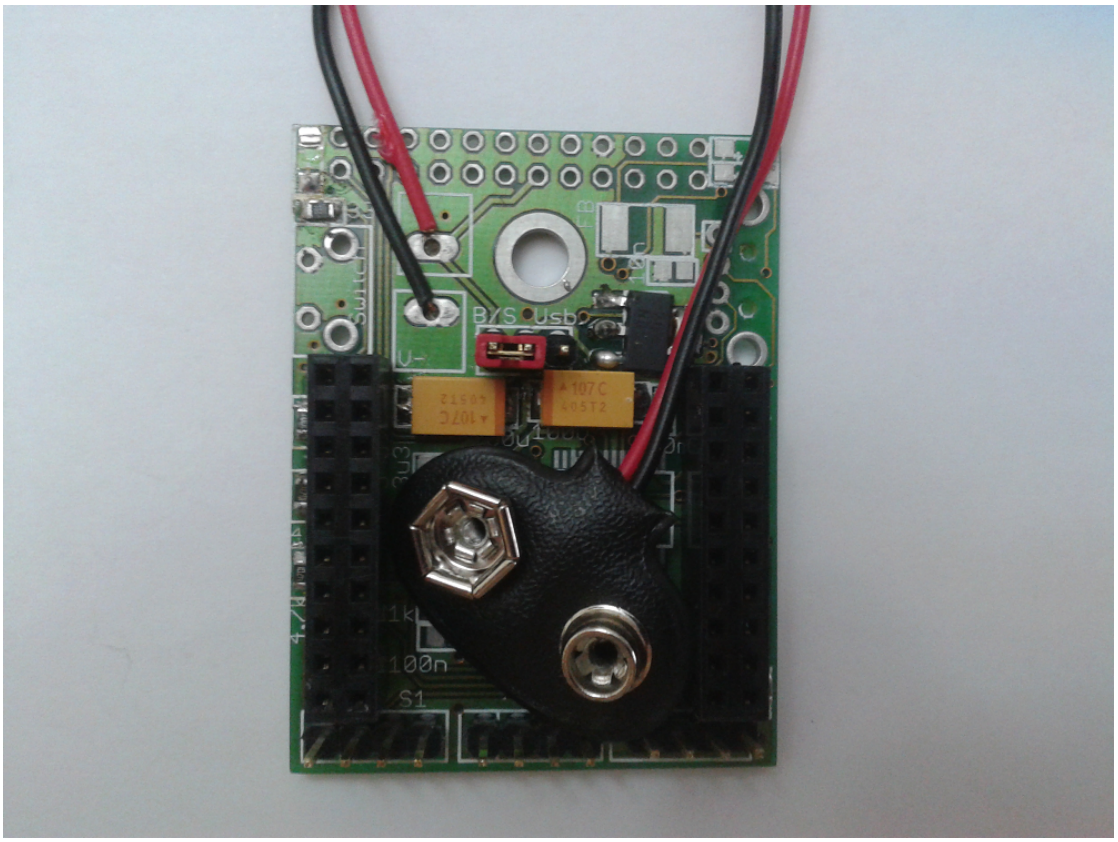


Figura 4.2 Placa em sua funcionalidade de Dispositivo-Final.

A placa visualizada na Figura 4.1.a é a placa que exerce a função de Coordenador/Gravadora. A Placa possui um circuito de tradução serial/usb que é usado para realizar a conexão direta com o computador que é necessária para receber programas e comandos e para enviar dados e resultados. Além deste circuito a placa tem um botão para reset utilizado no processo de gravação do programa, LEDs de indicação de energia de transmissão serial e também o regulador de tensão de 3,3V para a correta alimentação do ZigBit.

A placa de Dispositivo-Final, vista na Figura 4.1.b, possui também o LED de indicação de energia e o regulador de tensão, mas conta também com um divisor de tensão utilizado para a medição da bateria. A saída do divisor está conectado ao pino BAT do ZigBit que é utilizado na medição da carga da bateria.

O Esquemático da placa ZigBit Channel 1.0 se encontra no “Anexo V”.

Além da placa ZigBit Channel 1.0 utilizou-se uma outra placa que se anexava à ZigBit Channel 1.0 projetada pelo Engenheiro Felipe Brandão. É nessa placa que o Zigbit é acoplado com alguns componentes passivos necessários para o seu funcionamento. Na figura 4.2 abaixo, visualiza-se esta placa através de uma foto.

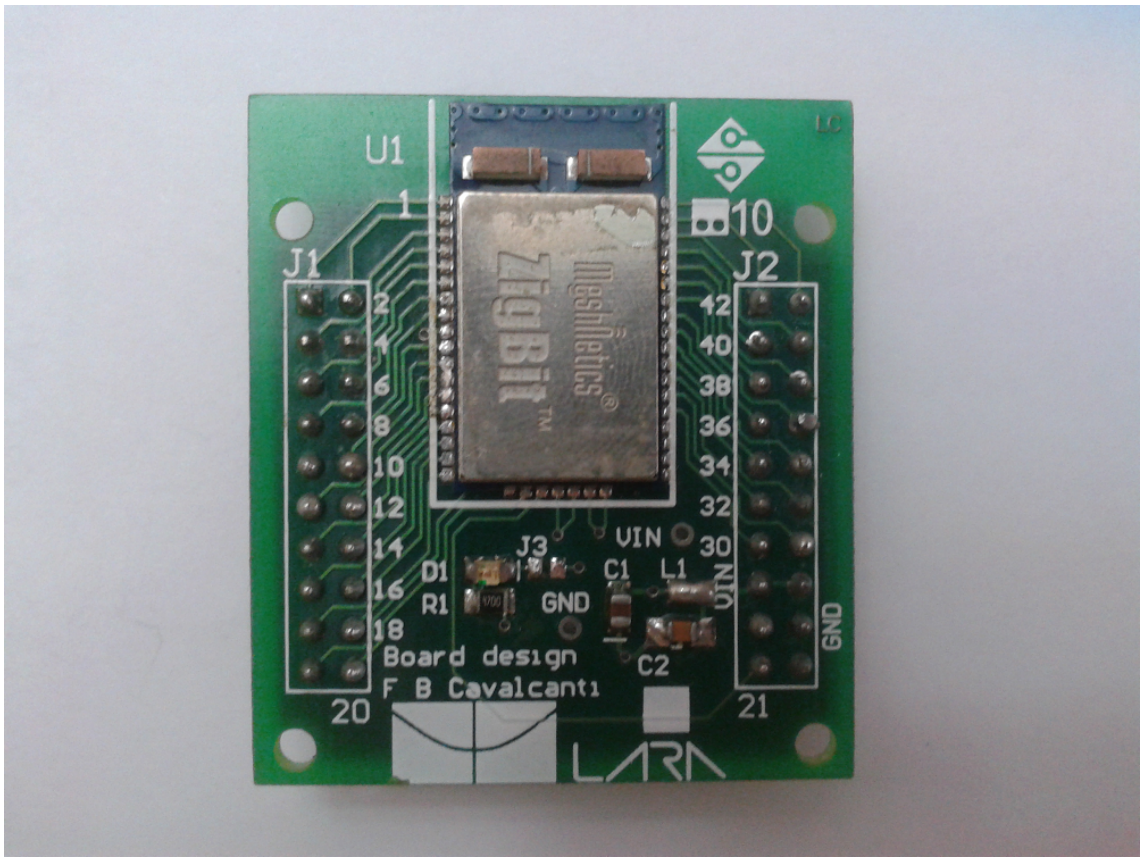


Figura 4.3 Placa ZigBit Breakout.

A placa ZigBit Breakout foi projetada pelo aluno da Universidade de Brasília, Felipe B Cavalcanti, que possui os direitos de fabricação da placa. O esquemático da placa não foi, portanto, fornecido para fins de estudo.

Para a implementação da rede, utilizamos a plataforma já fornecida que é a pilha BitCloud. A topologia da rede utilizada para os experimentos desta seção pode ser visualizada na Figura 4.3 abaixo.

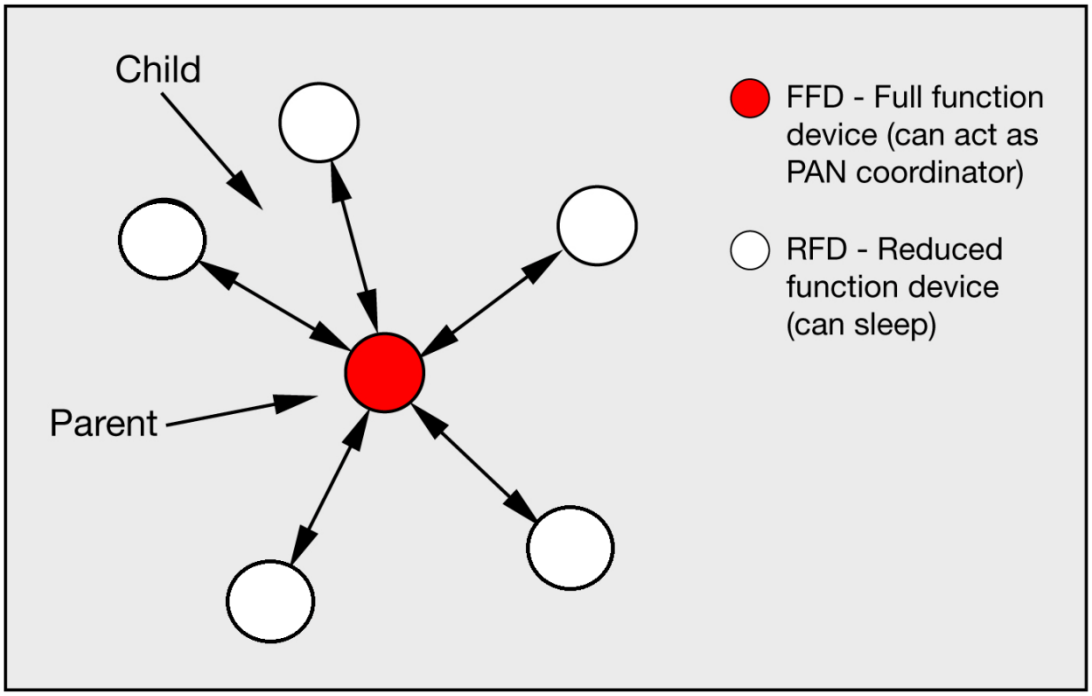


Figura 4.4 Topologia Utilizada na Rede. [15]

Em realidade a rede possuiu, no experimento 1, 6 sensores, e no experimento 2, 4 sensores. O que o leitor deve entender desse figura é que os Dispositivos-Finais da rede estavam todos ligados a um único coordenador que envia as informações para o computador.

4.2.2 ESTRUTURAÇÃO DA REDE

A função do nó coordenador, no caso do experimento, é simplesmente ativar o processo de medição e repassar ao computador todos os dados recebidos. É no coordenador que a rede toda se referencia, pois é ele que inicia e mantém a rede.

A programação do coordenador é muito simples. Inclui funções de inicialização de rede que determinam os parâmetros da rede que será criada, funções de comunicação wireless, que no caso do coordenador servem basicamente para receber os dados, e funções de comunicação serial que são utilizadas para receber comandos do computador ou para enviar os dados recebidos pela rede ao computador. A programação comentada do coordenador se encontra nos “Anexo I”. O algoritmo de funcionamento do coordenador pode ser verificado no diagrama de blocos mostrado abaixo na Figura 4.4.

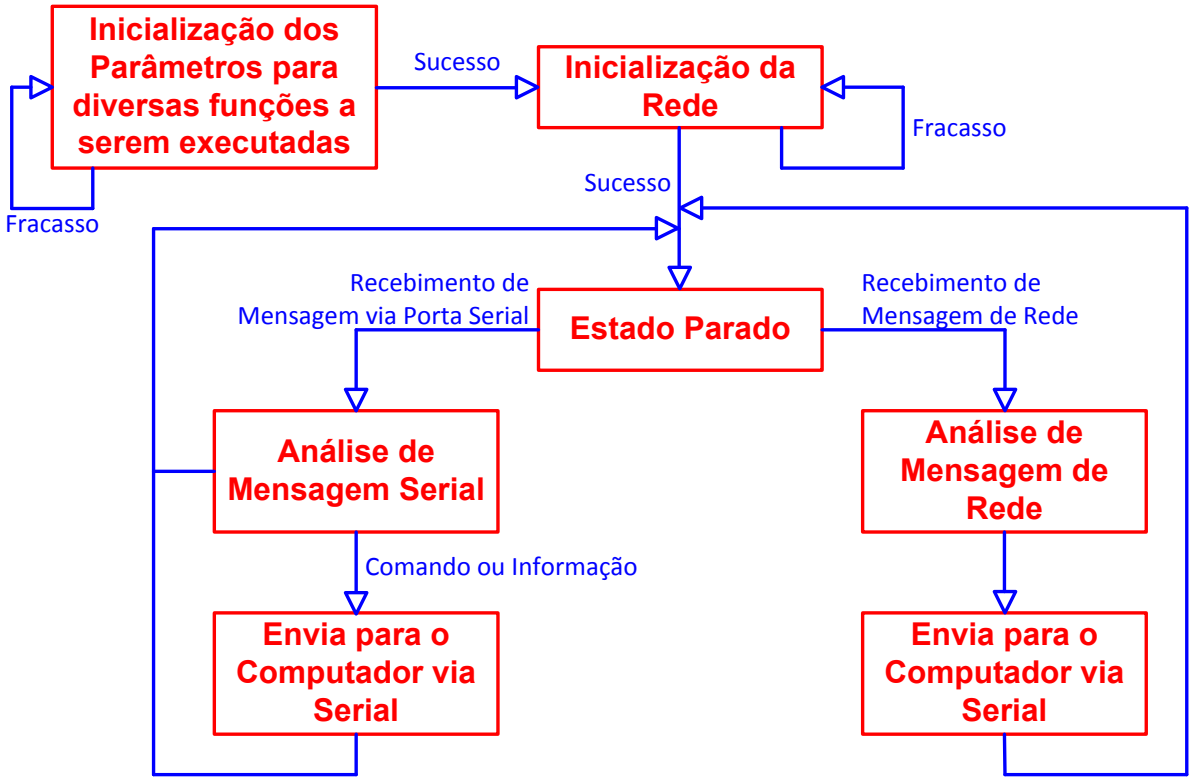


Figura 4.5 Fluxograma do Funcionamento do Coordenador.

A função do nó Dispositivo-Final é simplesmente enviar periodicamente os dados de energia ao coordenador. O DuCy do nó deve ser calculado previamente para que as temporizações sejam programadas corretamente.

A programação do nó Dispositivo-Final é também simples, mas possui mais funcionalidades que a programação do Coordenador. Possui as mesmas funções de inicialização de rede onde configura os parâmetros da rede à qual tentará se conectar. É importante que ao programar o ZigBit, tenha-se conhecimento dos parâmetros da rede para que a conexão seja feita com sucesso. O Dispositivo-Final também utiliza as funções de comunicação wireless, que são utilizadas para o envio de dados ao coordenador, não possui funções de comunicação serial, visto que não se conecta com o computador. Além dessas funções básicas o Dispositivo-Final também possui funções de inicialização de parâmetros de temporização, conversão analógico para digital e de protocolos de baixo gasto de energia, e claro, as funções que inicializam estas funcionalidades. O Programa comentado do Dispositivo-Final se encontra nos anexos sob o título de “Anexo II”. O algoritmo de funcionamento do Dispositivo-Final pode ser verificado no diagrama de blocos da Figura 4.5 abaixo.

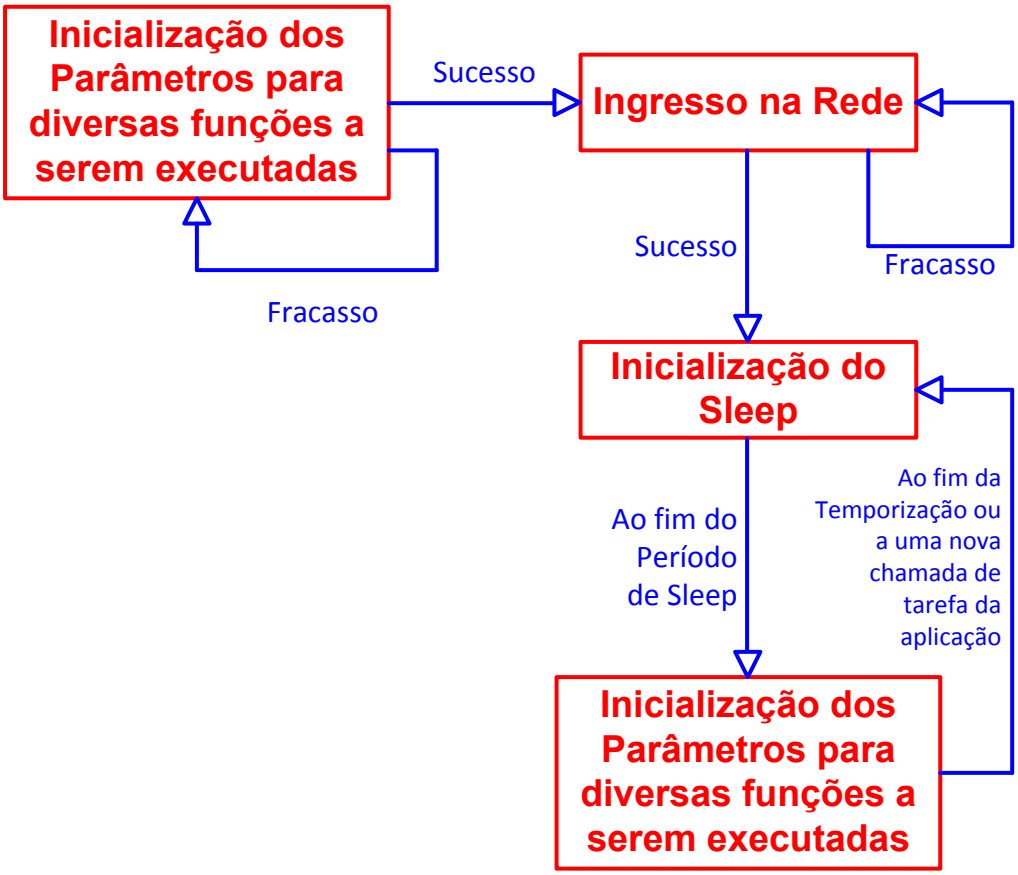


Figura 4.6 – Fluxograma do Funcionamento do Dispositivo-Final.

Para a Gravação dos módulos, utilizou-se o programa Serial Boot Loader, fornecido pela própria Atmel.

4.3 SOLUÇÃO PROPOSTA

Para propor uma solução, deve-se primeiramente analisar o que de fato está gerando o problema, ou seja, os pontos onde há o maior gasto de energia. Uma vez que os focos de gasto de energia sejam encontrados, é possível trata-los, de modo a reduzir a energia consumida, ou até mesmo neutralizá-los.

A análise foi feita primeiramente (e até o momento, exclusivamente) em nível de software. O ZigBit em seu funcionamento padrão utiliza por volta de 25mA. No evento de transmissão, o ZigBit utiliza por volta de 20mA a mais, e no evento de recepção, entre receber o dado e decodifica-lo o ZigBit utiliza aproximadamente 15mA de corrente. Os dois dados relativos aos eventos de transmissão e recepção foram retirados do *DataSheet* do ZigBit fornecido pela Atmel, já o dado do funcionamento sem eventos foi verificado de maneira experimental.

Os eventos de transmissão e recepção não podem ser neutralizados visto que são necessários para o funcionamento da rede, mas o gasto do ZigBit, quando não há eventos se deve pelo fato de que os dispositivos de transmissão estão ligados. A primeira solução então a ser implementada é desativar os dispositivos de transmissão e recepção wireless para que a utilização de corrente seja diminuída quando estes dispositivos estiverem sem utilização.

Dos motivos pelos quais escolhemos o dispositivo Zigbit, o que mais nos motivou foi a plataforma de desenvolvimento que a Atmel fornece através da pilha de programação, a BitCloud. Ao pesquisar a pilha a respeito de informações sobre economia de energia, foi encontrada, como já esperado, a função de Dormir. *Sleep*, do inglês, dormir, é uma funcionalidade muito comum de diversos controladores. A ideia geral do Dormir, embora varie de dispositivo para dispositivo, é desativar os canais de transmissão e muitas vezes também os de recepção, reduzindo o consumo do dispositivo ao consumo interno do controlador. No caso do ZigBit não é diferente. Ao implementar a função de Dormir, o ZigBit desativa seus canais de transmissão wireless e serial. O desligamento do dispositivo de recepção pode ou não ocorrer, mas cabe ao programador decidir e inserir uma única linha de código, visualizada abaixo:

```
#define CS_RX_ON_WHEN_IDLE false
```

Desta maneira, ao entrar no estado de Dormir, ou referenciado como *Idle*, do inglês, Inativo, o dispositivo também desativa o receptor wireless, reduzindo a corrente consumida a, de acordo com o *DataSheet* do produto, 6uA.

A primeira solução proposta se resume então a aplicar funcionalmente o protocolo de Dormir, o que significa que os dispositivos devem entrar em modo Dormir mesmo participando de uma rede e trocando dados com os outros dispositivos.

Outro fator que influencia no gasto de energia dos dispositivos ZigBit é a má utilização de seus recursos. Entenda-se por má, a utilização errônea e desorganizada de seus recursos. Como exemplo, o ADC. Ao ativá-lo, mesmo que não esteja em uso, começa a consumir carga. A segunda proposta para redução do consumo de energia é a boa prática de programação de desativar os periféricos como o ADC quando estes não estejam sendo utilizados.

Finalizam-se então as propostas para redução de energia em nível de software. Dentro da implementação proposta inicialmente é suficiente programar os Dispositivos-Finais com o protocolo de Dormir, verificar a funcionalidade do mesmo e comparar o gasto de energia de uma ZigBit que funciona sem o protocolo e entre outro que funciona com o protocolo.

4.4 EXPERIMENTO 1

Abaixo, na figura 4.7, encontra-se o esquemático com a planta baixa do laboratório LAVSI/LARA e onde foram posicionados cada um dos dispositivos utilizados no experimento.

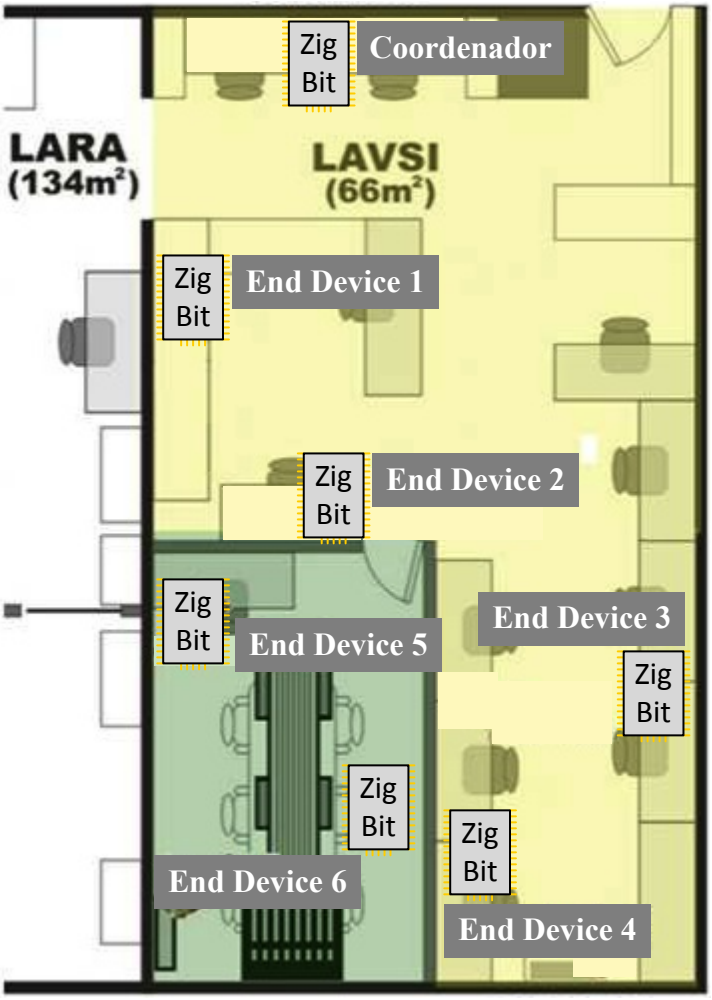


Figura 4.7. Planta baixa LAVSI/LARA com posicionamento dos dispositivos.

É possível perceber que as características de rede são do tipo estrela, pois cada um dos módulos sensores se conecta diretamente ao coordenador.

O primeiro experimento realizado no âmbito da economia de energia consistiu em criar-se uma rede na qual 6 Dispositivos-Finais estarão conectados a um coordenador enviando dados periodicamente. A diferença entre esses 6 dispositivos é que o Sensor1 e o Sensor2 não possuem nenhum protocolo para economia de energia, o Sensor3 e o Sensor4 dormem com um ciclo de 5 segundos, e os Sensores 5 e 6 dormem com ciclo de 10 segundos. Note que ainda não estão sendo realizadas comparações em relação ao DuCy, nos dois casos em que os dispositivos dormem, o DuCy tende a zero, pois o tempo de transmissão é a da ordem de poucos milissegundos, que perto da grandeza do ciclo não representam tempo significativo.

A programação do dispositivo-final neste experimento consistiu em dormir e ao acordar mandar um dado do seu ADC que refletia a carga da bateria. Assim que o dado era enviado, o dispositivo voltava a dormir.

Como o coordenador, no caso do experimento somente recebe mensagens e troca informações com o computador ao qual está acoplado, não existem nenhum tipo de tratamento em relação a envio de dados aos Dispositivos-Finais que não forem recebidos, visto que nenhum dado é direcionado a eles. Caso os Dispositivos-Finais não consigam enviar dados ao coordenador,

eles repetem o envio 5 vezes, e caso ainda assim não seja possível o envio, saem da rede e se conectam novamente para continuar a transmitir.

Na Tabela 4.1 abaixo, encontram-se a tensão inicial e final da bateria de cada sensor assim como o tempo de funcionamento aproximado de cada um. As baterias usadas no experimento são do modelo MN1604 da Duracell com tensão de 9V. O *DataSheet* desta bateria encontra-se nos no "Anexo IV".

Sensor	Tensão Inicial da Bateria	Tensão Final da Bateria	Tempo Aproximado de Funcionamento
1	9,34 V	6,24 V	11,81 h
2	9,15 V	6,01 V	11,47 h
3	9,19 V	5,10 V	43,64 h
4	9,53 V	5,56 V	40,55 h
5	9,33 V	4,81 V	45,81 h
6	9,45 V	4,80 V	48,00 h

Tabela 4.1 – Dados de Tensão e Tempo Estimado dos Dispositivos do Experimento 1.

O programa utilizado para coletar os dados do experimento foi o CoolTerm, um software grátis cujo download pode ser encontrado no link http://freeware.the-meiers.org/. Com este programa é possível enviar os dados recebidos para um arquivo de texto e assim tê-los salvos para depois trata-los de maneira mais confortável.

Pode-se verificar a partir da Tabela 4.1 que os dispositivos que não dormiram tiveram uma vida útil de bateria bem menor do que os dispositivos nos quais o protocolo foi aplicado. Note ainda que os dispositivos que enviaram menos mensagens, por terem um ciclo maior também tiveram um gasto de bateria menor, como era esperado.

No entanto, os tempos de duração de bateria encontrados foram bem menores do que os esperados, o que culminou em esforços de raciocínio ainda maiores sobre o problema. Com certeza alguma outra coisa consumia energia, mas como não foi encontrado nenhum motivo significante que explicasse esse consumo em nível de software, então, analisou-se o hardware.

No hardware percebeu-se que o que havia drenado corrente acima do esperado. 2 LEDs de indicação de energia, um em cada uma das placas do dispositivo, drenavam por volta de 8,0mA, que na grandeza da energia que estamos avaliando, tem participação significativa. Somente os LEDs não explicam, ainda, o consumo excessivo de energia, alguns mA ainda estavam sendo consumidos além do esperado. Verificou-se, então, que o regulador de tensão era o responsável. No projetar da placa, não houve percepção de que a energia consumida pelo regulador, mesmo sendo tão baixa em outros sistemas, na situação avaliada, com um consumo aproximado de 5,7mA, também representava uma carga significativa.

Na Tabela 4.2 abaixo é possível verificar os dados exatos de cada um desses parâmetros para cada um dos sensores.

Sensor	Corrente da Placa (Regulador de Tensão)	Corrente Dos LEDs	Corrente Total.
1	7,70 mA	6,00 mA	38,18 mA
2	7,70 mA	6,00 mA	38,19 mA
3	7,93 mA	5,68 mA	13,63 mA
4	7,89 mA	5,71 mA	13,62 mA
5	7,63 mA	5,71 mA	13,34 mA
6	5,67 mA	7,52 mA	13,20 mA

Tabela 4. 2 – Corrente dos Dispositivos do Experimento 1.

Com esses dados à mão, é possível realizar um cálculo de estimação do tempo de funcionamento do ZigBit sem os empecilhos do hardware. O processo de cálculo se dá da seguinte maneira:

Inicialmente verificamos a carga útil total desprendida pela bateria. Verifique na Figura 4.6 abaixo, tirada do *Datasheet* da bateria [13], que a carga útil da bateria varia de maneira diretamente proporcional com a carga média à qual é submetida, ou seja, nos sensores sem o procedimento de Dormir, espera-se uma carga útil menor que nos sensores que possuem o procedimento de Dormir.

De acordo com as correntes medidas durante o funcionamento do ZigBit, verificamos que a curva à qual os dispositivos 1 e 2 se aproximam é a curva de 1000mW e os outros dispositivos se aproximam da curva de 250mW. Todos os dispositivos utilizaram a mesma bateria.

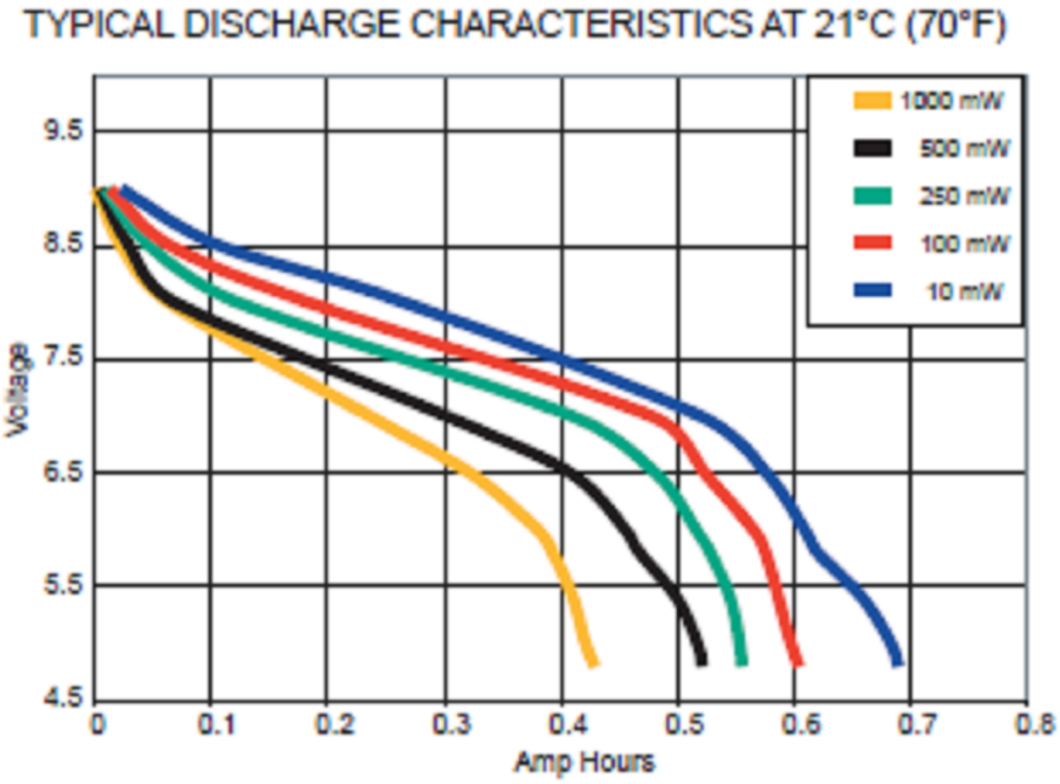


Figura 4.8. – Carga útil da bateria em função da potência. [13]

Com esses dados em mão, podemos começar um cálculo de estimativa de tempo de duração do ZigBit caso o dispositivo não tivesse o gasto dos LEDs nem o do regulador de tensão. Para tal, temos que realizar uma série de cálculos.

Primeiramente realizamos o Calculo de Carga Total.

$$CargaTotal = CorrenteTotal * Tempo \quad (1)$$

Em seguida calculamos a carga consumida nos LEDs e no regulador de tensão.

$$CargaLEDs = Tempo * CorrenteLEDs \quad (2)$$

$$CargaReg = Tempo * CorrenteReg \quad (3)$$

Com cada carga bem determinada o próximo passo é calcular a carga total consumida apenas pelo ZigBit.

$$CargaZigBit = CargaTotal - CargaReg - CargaLEDs \quad (4)$$

E com o dado da carga consumida pelo ZigBit, a corrente média que o mesmo consumiu, não só para verificar o dado medido anteriormente, mas porque o cálculo é mais preciso desta maneira.

$$CorrenteZigBit = \frac{CargaZigBit}{Tempo} \quad (5)$$

E finalmente, é possível calcular o tempo estimado de duração do ZigBit caso se retirasse os LEDs e o regulador de tensão.

$$TempoEstimado = \frac{CargaTotal}{CorrenteZigbit} \quad (6)$$

Note que, de acordo com a figura 4.7, que nos mostra que para uma menor potência a carga útil da bateria é maior, a carga útil total provavelmente seria maior, ou seja, o tempo estimado de funcionamento do ZigBit seria ainda maior do que o calculado desta maneira.

Na Tabela 4.3 abaixo, verificamos os valores calculados de cada um dos parâmetros descritos acima para cada um dos dispositivos utilizados no experimento.

Sensor	Carga Total	Carga do Regulador de Tensão	Carga dos LEDs	Carga do ZigBit	Corrente no ZigBit	Estimativa de Funcionamento
1	451 mAh	70,86 mAh	90,94 mAh	289,57 mAh	24,52 mA	18,39 h
2	438 mAh	68,82 mAh	88,43 mAh	280,77 mAh	24,47 mA	17,89 h
3	594,8 mAh	247,87 mAh	346,1 mAh	0,88 mAh	20,16 µA	>29000 h ou >1200 h ou >3 Anos
4	552,3 mAh	231,54 mAh	320,75 mAh	0.4 mAh	9,8 µA	>50000 h ou >2000h ou >5 anos
5	611,56 mAh	261,57 mAh	349,53 mAh	0.47 mAh	10,10 µA	>50000 h ou >2000 h ou > 5 Anos
6	633,6 mAh	271,58 mAh	360,96 mAh	0.98 mAh	20,41 µA	>30000 h ou >1200 d ou >3 Anos

Tabela 4.3 – Estimativa de duração da bateria excluindo-se LEDs e Regulador de Tensão.

Obviamente, pela precisão dos dados medidos e pelas condições de como o experimento foi feito, a aproximação ainda é grosseira para os módulos que possuem o procedimento de Dormir. Note também que o valor de DuCy é virtualmente 0%, ou tende a 0%, o que também não é muito adequado para uma boa análise do protocolo. Faz-se necessário um novo experimento que tenha menos interferência de carga, e que trabalhe com ciclos específicos de Dormir para analisarmos melhor o comportamento do protocolo.

4.5 EXPERIMENTO 2

O segundo experimento realizado no âmbito da economia de energia visa aplicar outra proposta além da inicial de se reduzir o gasto apenas via software. Baseado nos resultados e medidas anteriores, a terceira proposta para a economia de energia eficiente nos dispositivos é retirar os LEDs e o regulador de tensão, mesmo que retirar o regulador de tensão traga consigo o problema de falta de flexibilidade no uso de baterias, pois restringe o uso de baterias a algumas específicas. Como podemos ver no *DataSheet* do regulador utilizado, que se encontra nos anexos sob o título de Anexo K, seu funcionamento está na faixa de XD – XD V, proporcionando uma ampla gama de baterias que poderiam ser utilizadas. Mesmo assim, ao analisar a situação, consideramos que é melhor estar restrito a algumas baterias do que ao alto consumo do regulador de tensão. Infelizmente, não houve tempo hábil para a aquisição de baterias que suprissem a necessidade que tínhamos em relação à alimentação do ZigBit, por isso, o segundo experimento foi realizado ainda com os reguladores, mas sem os LEDs.

A grande diferença entre o Experimento 1 e o Experimento 2, é que agora os sensores vão trabalhar com um DuCy específico como mostrado na Figura 4.X abaixo.

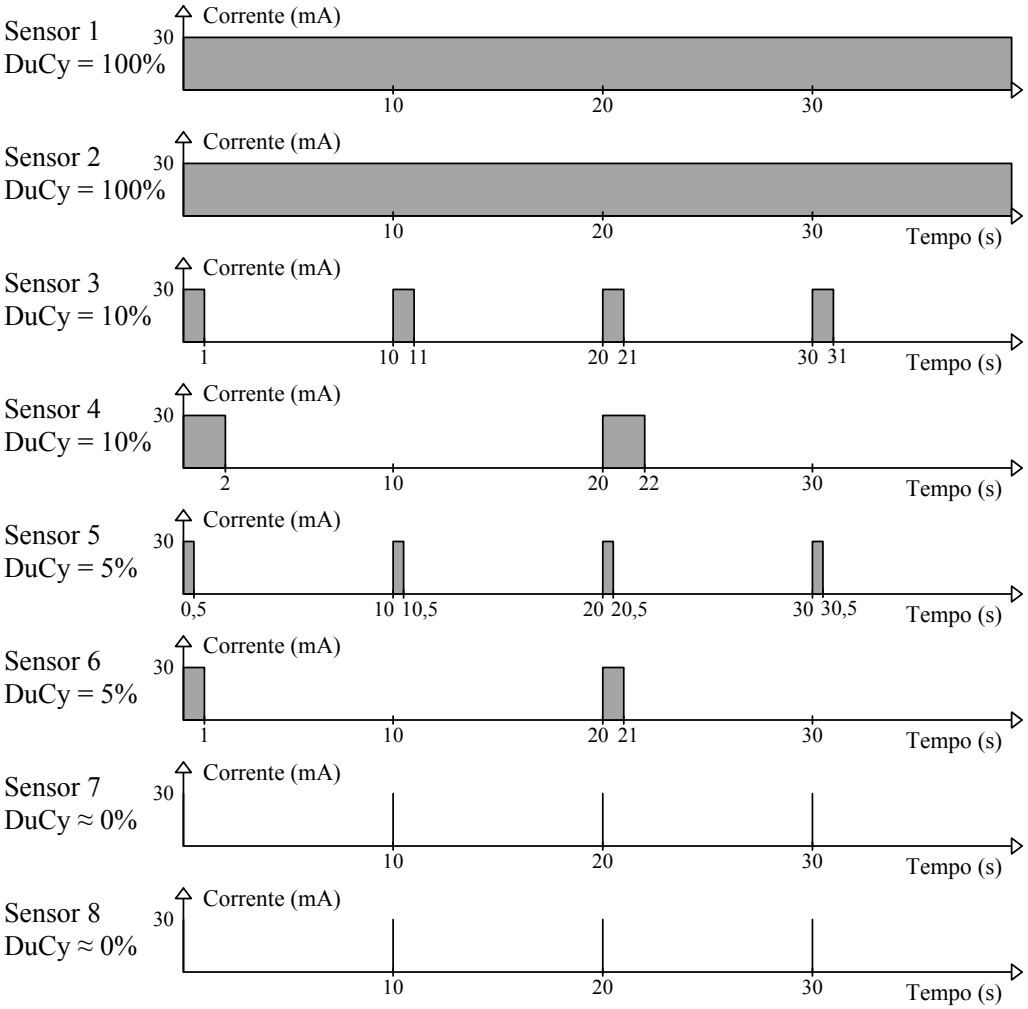


Figura 4.9. Duty Cycles por sensor para o Experimento 2.

O experimento 2 foi realizado com apenas 4 ZigBits diferentes na realidade, em 2 momentos. Primeiramente funcionando como Sensor1 a Sensor4 e depois funcionando como Sensor5 a Sensor8.

Na Tabela 4.4 abaixo, temos a tensão inicial e final da bateria de cada sensor assim como o tempo de funcionamento aproximado de cada um. As baterias usadas no experimento são do modelo MN1604 da Duracell com tensão de 9V.

Sensor	Tensão Inicial da Bateria	Tensão Final da Bateria	Tempo Aproximado de Funcionamento
1	9,53 V	5,691 V	16,71 h
2	9,51V	5,659 V	16,20 h
3	9,51 V	5,150 V	62,45 h
4	9,53 V	5,854 V	58,15 h
5	9,52 V	4,816 V	77,31 h
6	9,52 V	4,810 V	93,67 h
7	9,53 V	5,971 V	103,44 h
8	9,51 V	5,987 V	105,52 h

Tabela 4.4 – Dados de Tensão e Tempo Estimado dos Dispositivos do Experimento 2.

Neste experimento o regulador de tensão é o único dispositivo que drena corrente significativa além do ZigBit. Abaixo temos na Tabela 4.5 os valores das correntes em cada dispositivo.

Sensor	Corrente da Placa (Regulador de Tensão)	Corrente Total Média
1	5,726 mA	30,71 mA
2	5,694 mA	30,64 mA
3	5,726 mA	8,279 mA
4	5,694 mA	8,942 mA
5	5,688 mA	6,923 mA
6	5,659 mA	6,952 mA
7	5,688 mA	5,703 mA
8	5,659 mA	5,680 mA

Tabela 4.5 –Corrente dos Dispositivos do Experimento 2.

O processo de calculo para estimarmos o tempo é o mesmo que utilizamos no experimento 1, ainda em vista do que podemos atestar na figura 4.6. A única diferença nos cálculos reside na fórmula [4], pois não há carga gasta por LEDs.

Primeiramente realizamos o Calculo de Carga Total.

$$CargaTotal = CorrenteTotal * Tempo \quad (7)$$

Em seguida calculamos a carga consumida nos LEDs e no regulador de tensão.

$$CargaLEDs = Tempo * CorrenteLEDs \quad (8)$$

$$CargaReg = Tempo * CorrenteReg \quad (9)$$

Com cada carga bem determinada o próximo passo é calcular a carga total consumida apenas pelo ZigBit.

$$CargaZigBit = CargaTotal - CargaReg \quad (10)$$

E com o dado da carga consumida pelo ZigBit, a corrente média que o mesmo consumiu, não só para verificar o dado medido anteriormente, mas porque o cálculo é mais preciso desta maneira.

$$CorrenteZigBit = \frac{CargaZigBit}{Tempo} \quad (11)$$

E finalmente, é possível calcular o tempo estimado de duração do ZigBit caso se retirasse os LEDs e o regulador de tensão.

$$TempoEstimado = \frac{CargaTotal}{CorrenteZigbit} \quad (12)$$

Novamente, de acordo com a Figura 4.6, nesse caso, a carga útil total provavelmente seria maior, ou seja, o tempo estimado de funcionamento do ZigBit seria ainda maior do que o calculado desta maneira.

Na Tabela 4.6 abaixo, verificamos os valores calculados de cada um dos parâmetros descritos acima para cada um dos dispositivos utilizados no experimento.

Sensor	Carga Total	Carga do Regulador de Tensão	Carga do ZigBit	Corrente no ZigBit	Estimativa de Funcionamento
1	513.03 mAh	95.68 mAh	417.34 mAh	24.97 mA	20.94 h
2	496.46 mAh	92.23 mAh	404.20 mAh	24.94 mA	19.91 h
3	517.02 mAh	357.59 mAh	159.43 mAh	2.553 mA	202.52 h (8.43 d)
4	519.97 mAh	331.11 mAh	188.86 mAh	3.248 mA	160.08 h (6.67 d)
5	535.21 mAh	439.74 mAh	95.47 mAh	1.234 mA	433.40 h (18.1 d)
6	651.19 mAh	530.08 mAh	121.11 mAh	1.293 mA	503.64 h (21.0 d)
7	589.92 mAh	588.37 mAh	1.5533 mAh	15.01 µA	39285 h (4.48 anos)
8	599.34 mAh	597.14 mAh	2.2023 mAh	20.87 µA	28716 h (3.27 anos)

Tabela 4.6 - Estimativa de duração da bateria excluindo-se o Regulador de Tensão.

.

É possível a partir da Tabela 4.6 verificar o que havia sido colocado antes e mostrado na Figura 4.6 retirada do *DataSheet* da bateria utilizada. Com a diminuição da corrente, ou seja, da potência média utilizada, a carga útil da bateria é maior. Note na Tabela 3 que os valores de carga útil para cada dispositivo são menores, assim como a estimativa de tempo para o caso dos Sensores 1 e 2. Dessa forma, é possível afirmar, agora com mais certeza, que na verdade a estimativa de funcionamento retorna na verdade um valor menor do que o que seria na realidade.

O que podemos verificar também é que no caso dos dispositivos com o procedimento de Dormir implementado, existe grande diferença de tempo de duração. Isso ocorre porque ao contrário do caso dos sensores que não utilizam o protocolo e gastam a maior parte de sua carga com o próprio ZigBit, os que utilizam o protocolo gastam a maior parte de sua carga na própria placa ZigBit Channel 1.0, e como cada placa tem um gasto diferente, verificamos diferença na projeção de tempo.

Outro fator responsável por essa diferença são os próprios ZigBits, alguns, por motivos desconhecidos apresentam corrente muito superior a 6uA quando estão em modo Dormir, correntes na faixa de 3mA. Novos programa foram passados para o ZigBit, a placa ZigBit Channel foi trocada, mas sem resultado positivo.

4.6 CONSIDERAÇÕES FINAIS

Por fim, será realizada uma análise final nos dados dos sensores e da maneira de seu funcionamento.

4.6.1 DECAIMENTO DA BATERIA

Primeiramente será feita a análise do decaimento da bateria em cada um dos sensores utilizados. A análise será feita para confirma a robustez da rede e do envio correto de dados dos sensores ao coordenador, para verificar o comportamento da bateria em cada um dos DuCy e finalmente, não tão importante, verificar a veracidade dos dados do *DataSheet* da bateria.

Para a análise do decaimento da bateria serão utilizadas as medições feitas pelos ADC de cada dispositivo. A curva característica do ADC de cada dispositivo está mostrada na Figura 4.7 abaixo.

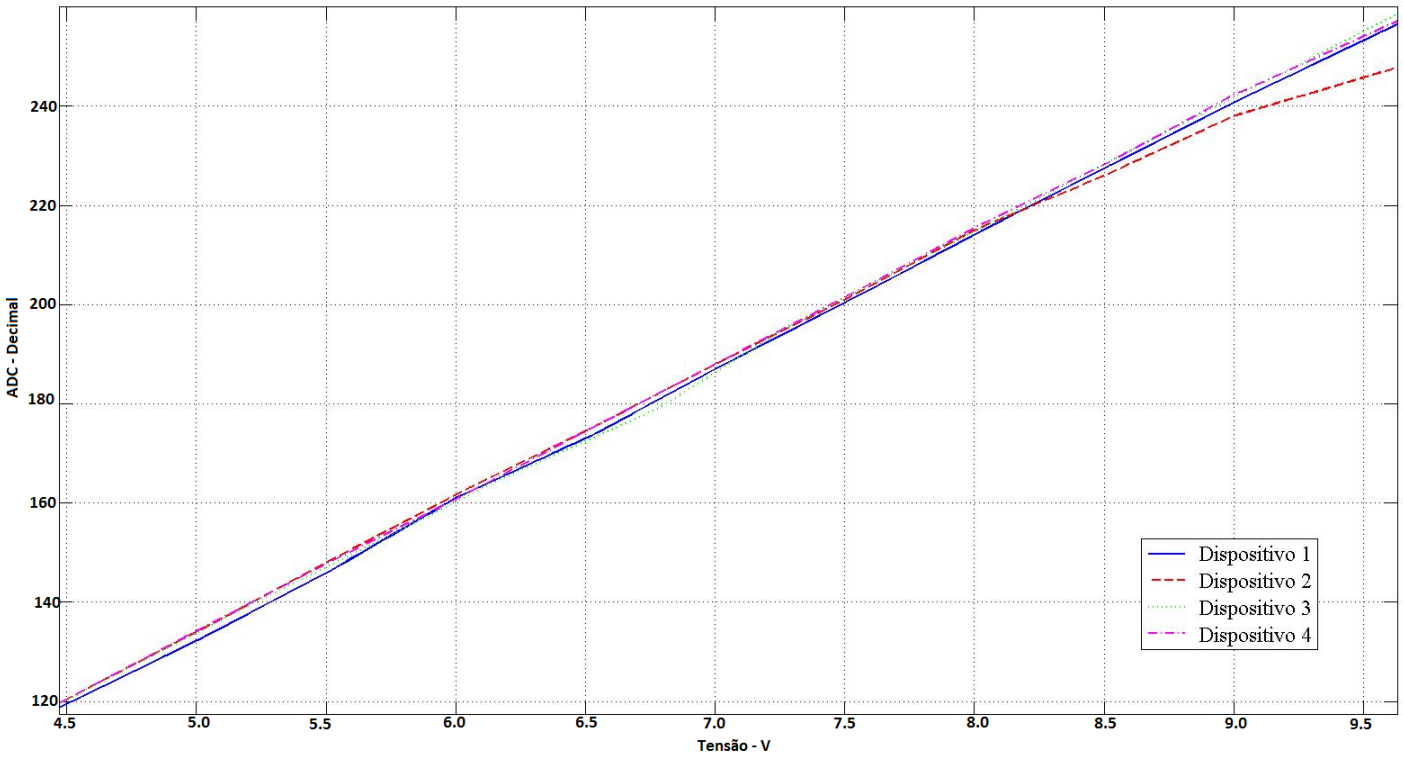


Figura 4.10. ADCs dos Dispositivos.

Lembrando que o Dispositivo 1 é tanto o Sensor1 quanto o Sensor3, o Dispositivo 2 é tanto o Sensor2 quanto o Sensor4 o Dispositivo 3 é tanto o Sensor5 quanto o Sensor7 e o Dispositivo 4 é tanto o Sensor6 quanto o Sensor8.

Os ADC de todos os dispositivos funcionam de maneira praticamente linear, então serão considerados como uma fonte confiável para se referenciar.

Os dados das medições de cada ADC foram enviados ao coordenador ao longo do tempo e salvos num documento de texto para serem tratados. Para que isso fosse realizado, utilizou-se

um programa de comunicação com porta serial chamado CoolTerm, já mencionado anteriormente.

Uma que os dados estavam disponibilizados num arquivo texto, utilizou-se um programa em C, compilado e executado no programa DevC++ de modo a extrair os dados e separá-los em diferentes arquivos, um arquivo para cada sensor. Isso foi feito para que se pudesse usar os arquivos no MatLab. O Programa que faz a separação dos dados se encontra no "Anexo III".

Utilizando os arquivos gerados anteriormente, podemos criar um vetor com os dados do arquivo no MatLab e criar também um vetor temporal gerado por um for, e , finalmente, plotar os gráficos de decaimento de bateria. Na Figura 4.8 Abaixo podemos verificar o decaimento da bateria de cada sensor. O tempo nos gráficos está em segundos e o valor da bateria está no valor decimal da conversão feita no ADC.

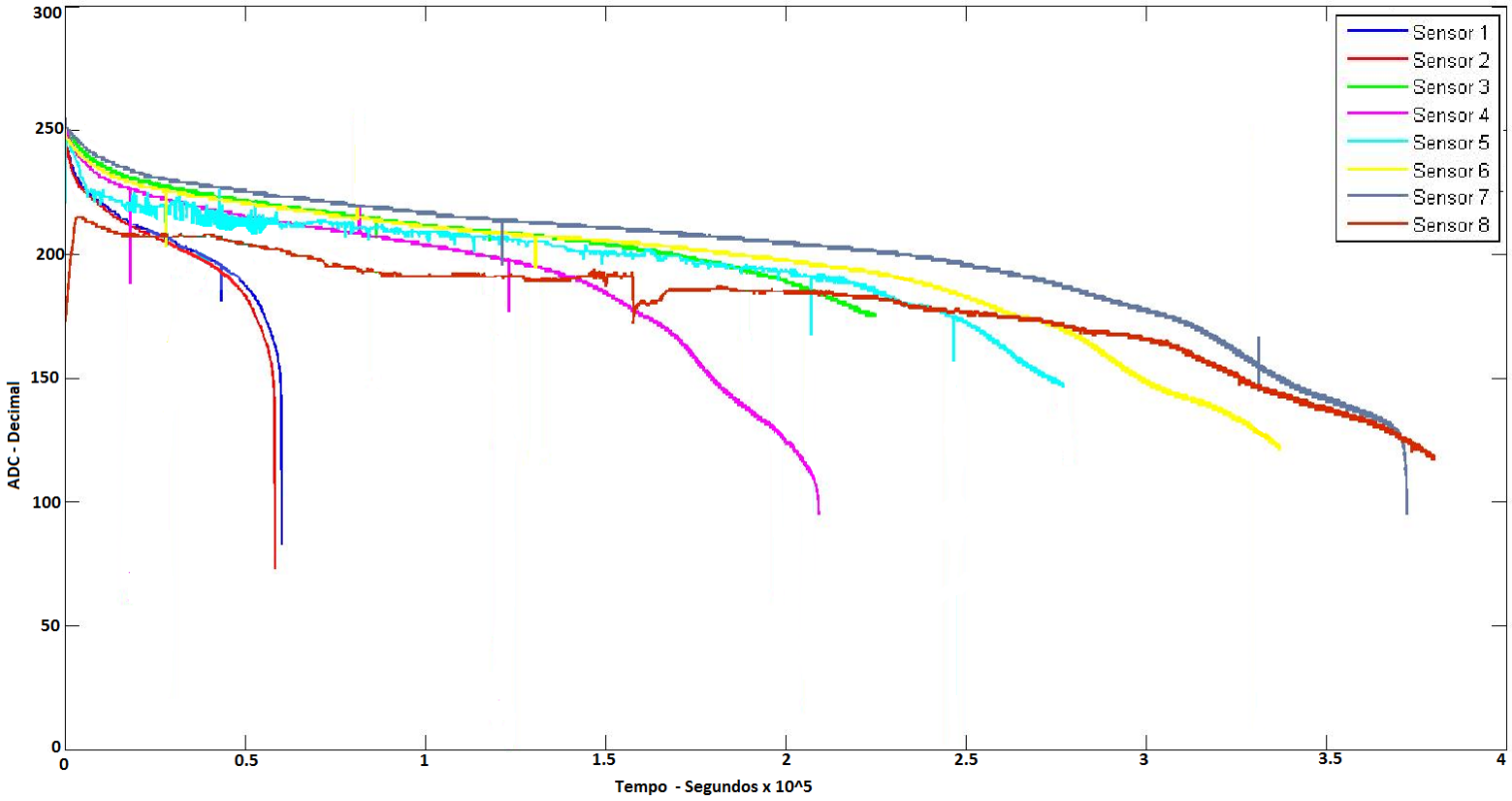


Figura 4.11. Decaimento da Bateria dos Sensores.

Antes de realizar a análise dos gráficos, é importante um consideração. Note que nas curvas acima possuem alguns erros as vezes um pico abaixo da curva continua ou acima. Tal evento ocorre por causa da oscilação da referência do ADC. Algumas vezes, poucas como podemos observar, o ADC ao ser iniciado não consegue em tempo hábil estabilizar a referência e por causa disso, algumas medidas são realizadas com a referência errada e por isso retornam um dado errado. É importante ressaltar que o número de dados coletados são sempre na casa dos milhares, e os erros visualizados nas curvas são sempre inferiores a um dezena, o que mostra que o número de erros é muito pequeno.

Já havia sido verificada anteriormente a relação da potência gasta com a carga útil da bateria. Agora, ao analisar estes gráficos é possível verificar como a fonte de energia limitada (pilhas ou baterias), se comporta ao longo do tempo de acordo com a corrente média utilizada pelo

dispositivo. Infelizmente em função de o regulador de tensão ter sido utilizado no experimento, não existem resultados com correntes médias menores que 5,7mA aproximadamente, e talvez por isso os dados não sejam suficientes para se fazer uma análise muito precisa e extremamente completa, no entanto, os dados possibilitam pelo menos uma análise não apenas superficial da situação.

Percebe-se que o decaimento é mais suave quanto menor for a corrente média no dispositivo, ou seja, quanto menor for o seu DuCy. É importante notar isso porque as soluções empregadas para a redução do consumo de bateria se provam eficientes nos gráficos acima e tabelas mostrados até agora neste capítulo, pois nota-se que a modificação do hardware claramente melhorou o desempenho da bateria e também que a aplicação de Dormir é a real responsável pela diminuição do consumo a nível de software.

Note finalmente a partir dos gráficos que, como esperado, quanto menor o DuCy do dispositivo, maior o tempo de duração da bateria do mesmo.

Uma observação factível é que os sensores que trabalharam com o mesmo DuCy apresentam tempo de funcionamento diferente um do outro e geralmente o que tem o gráfico de decaimento mais conturbado é o que durou menos. Isso ocorre porque algumas vezes o sensor acaba por se desconectar da rede, mas o programa possui um protocolo de recuperação de rede. No entanto, a recuperação de rede demanda um gasto de energia maior, pois quando o ZigBit tenta se conectar na rede sua corrente média supera os 20mA e como a reconexão à rede, pode as vezes demorar alguns minutos, o gasto de corrente ao final fica sendo maior.

4.6.2 DURAÇÃO DA BATERIA X CICLO DE TRABALHO

A análise final e talvez a mais importante que deve ser desenvolvida nesse trabalho, é relacionar o DuCy à duração da bateria. Para tal, utilizamos os dados calculados do Experimento 2 e os respectivos DuCy e podemos então plotar um outro gráfico de gasto de energia em função do DuCy. A Figura 3.9 abaixo mostra o comportamento da bateria em relação ao DuCy de acordo com os dados coletados do Experimento 2.

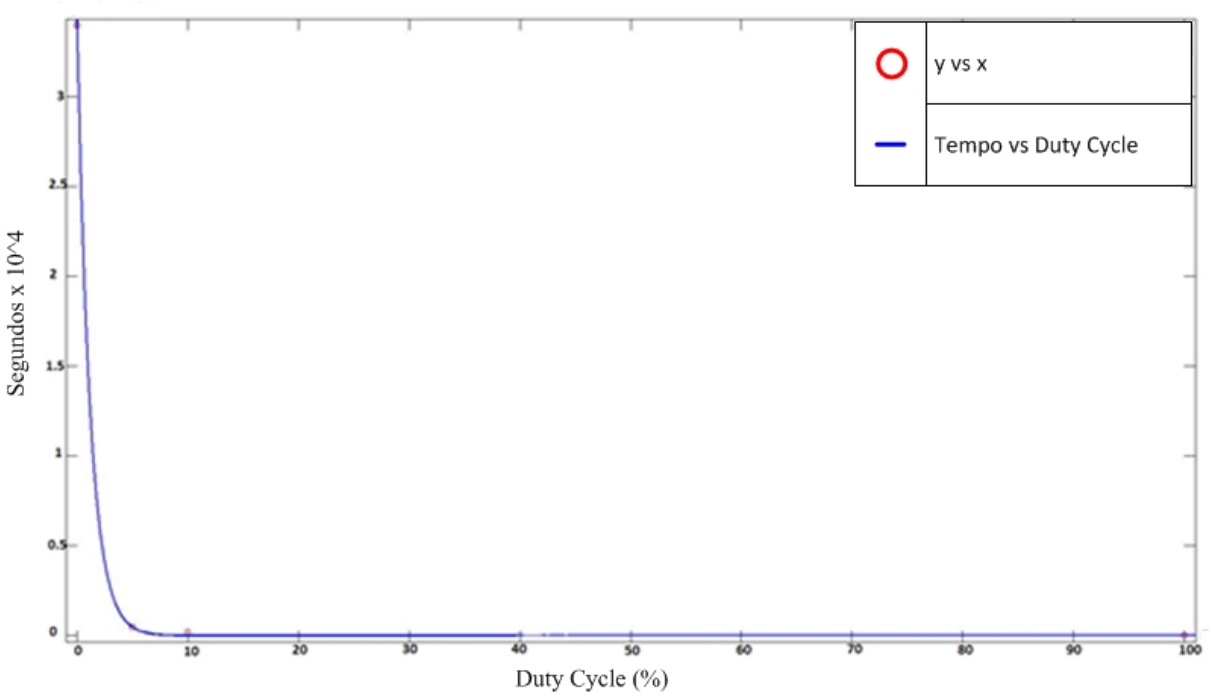


Figura 4.12. Tempo estimado de funcionamento por DuCy.

De acordo com a figura 4.20, podemos verificar que o ideal é reduzir o DuCy ao mínimo necessário para potencializar a utilização da bateria de diversos modos. Se baseando na figura 4.20, podemos verificar que mesmo que o ganho com o modo Dormir utilizando um certo DuCy seja significativo, a duração da bateria decai exponencialmente com o valor do DuCy, significando que mesmo um DuCy considerado pequeno como 10%, pode comprometer a utilização da bateria. Para reduzirmos o DuCy, devemos nos ater a apenas executar o estritamente necessário para o funcionamento da rede, para tal, utiliza-se uma função da BitCloud para *Polling* e o algoritmo do nó Dispositivo-Final é um pouco modificado. O algoritmo de funcionamento sugerido com base nos dados coletados é o visualizado na figura 4.21 abaixo.

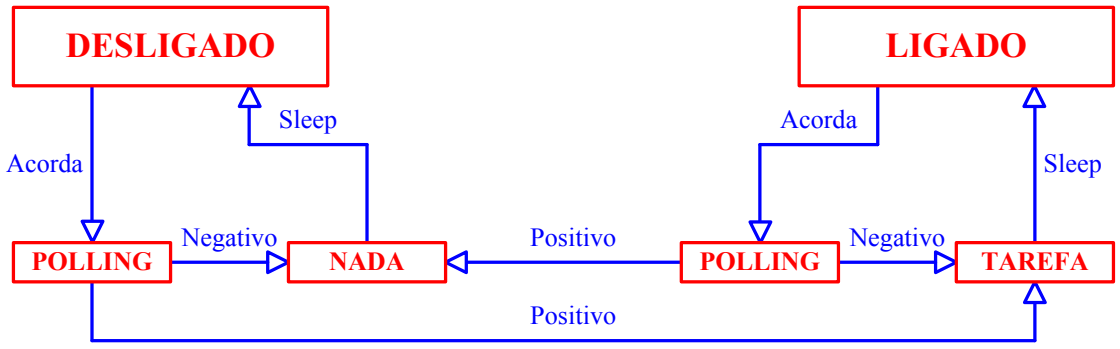


Figura 4.13. Máquina de Estados do sistema.

A filosofia básica deste algoritmo é fazer o *Polling* de mensagem, ou seja, verificar se há alguma mensagem direcionada para o nó em questão e caso haja, mudar (ou não mudar) o seu

estado de acordo com a mensagem recebida, e de acordo com o seu estado, realizar ou não certas tarefas. Na figura, denota-se por *Polling* Positivo se existe uma mensagem que gere mudança de estado no nó em questão, e por *Polling* Negativo se não houver nenhuma mensagem ou se a mensagem não gere mudança de estado do nó. Note também que ao finalizar a execução da tarefa o ZigBit volta ao modo Dormir ao invés de esperar algum tempo inativo. Pode-se acreditar que desta maneira fica difícil sincronizar o envio de mensagens, mas é por isso que foi utilizada a função de Polling ZDO_StartSyncReq.

Com esta última ponderação, encerra-se a análise a respeito do gasto de energia do ZigBit.

5 GRAVAÇÃO REMOTA SOBRE ZigBit™

No capítulo 5 deste relatório, serão abordados descrição do problema, implementação, dados e resultados, assim como análise dos dados coletados a respeito da gravação remota sobre módulos ZigBit.

Redes ZigBee são bastante complexas do ponto de vista de aplicação e de quantidade de módulos, especialmente na automação predial, que requer uma quantidade gigantesca de nós para monitoramento e controle de sua operação.

Diferentemente de redes convencionais, nas quais nós sensores e atuadores podem ser instalados a milhares de metros da controladora de rede sem a necessidade de nó roteador ou coisa do tipo, as redes wireless precisam de nós de expansão de rede, os roteadores, que têm a função de replicar as informações recebidas para o nó adiante nos dois sentidos (coordenador para sensor e sensor para coordenador). Mais especificamente nas redes ZigBee, onde o alcance pode ser menor do que o que se encontra em Wi-Fi, por exemplo, esse número pode ser ainda maior.

Em uma rede ZigBee, com uma quantidade de milhares de dispositivos, pode ser de uma complexidade enorme manter todos os nós trabalhando e cooperando uns com os outros da maneira como se espera. Como monitorar todos os módulos ZigBee da rede? Em caso de problemas com algum nó específico, como diagnosticar o problema e como solucioná-lo de uma maneira rápida e prática?

Neste intuito, surge, mais do que uma aplicação em busca de conforto, a necessidade da gravação remota. Isso quer dizer que cada módulo deve ter a possibilidade de ser atualizado sem estar diretamente ligado a um computador.

5.1 OVER-THE-AIR UPGRADE

A necessidade de se diagnosticar e resolver problemas de rede, bem como de atualizar funcionalidades de diversos nós de rede, remotamente fez com que os engenheiros de software da Atmel desenvolvessem a aplicação da funcionalidade da gravação remota dentro da pilha BitCloud. É o que se chama de Over-The-Air Upgrade (OTAU), traduzido no sentido literal como Atualização Sobre o Ar.

Esta funcionalidade permite ao programador colocar em qualquer nó da rede a possibilidade de se atualizar a imagem presente, substituindo-a por uma nova. Isso é feito com a intenção de se alterar sua funcionalidade ou de se solucionar um problema de mal funcionamento de um nó.

A principal aplicação dessa funcionalidade se encontra em casos onde se torna difícil o acesso aos nós, em geral por questões físicas, como ambientes de difícil acesso, ambientes desconfortáveis devido à temperatura (como casas de máquinas ou centrais de energia), dentre outros. Porém, pode ser utilizada simplesmente pelo conforto de não ser necessária a operação em campo, mas sim remota.

Levando-se em conta que uma rede ZigBee de monitoramento predial pode ter quantos nós se queira, a aplicação OTAU torna-se algo essencial à rede.

O suporte do OTAU em uma rede ZigBee é coberta pela especificação Over-The-Air Cluster, que é parte da pilha de bibliotecas BitCloud. Esta especificação se baseia no uso padrão da ZCL (ZigBee Cluster Library) de transferência de dados sobre a rede com o aditivo da possibilidade de atualização remota para qualquer nó da mesma.

Os componentes principais da arquitetura da aplicação OTAU são:

- O cliente OTAU, que são os nós finais de rede que receberão o novo firmware para atualização;
- O servidor OTAU, responsável pela transferência do pacote aos clientes e pelo processamento da gravação remota;
- Um dispositivo HAL, responsável por armazenar a imagem transferida em alguma memória persistente em cada dispositivo com possibilidade de gravação;
- Um Bootloader (software responsável por gravação) com aplicabilidade OTAU.

A arquitetura de um cliente OTAU é a que se mostra na figura 4.22.

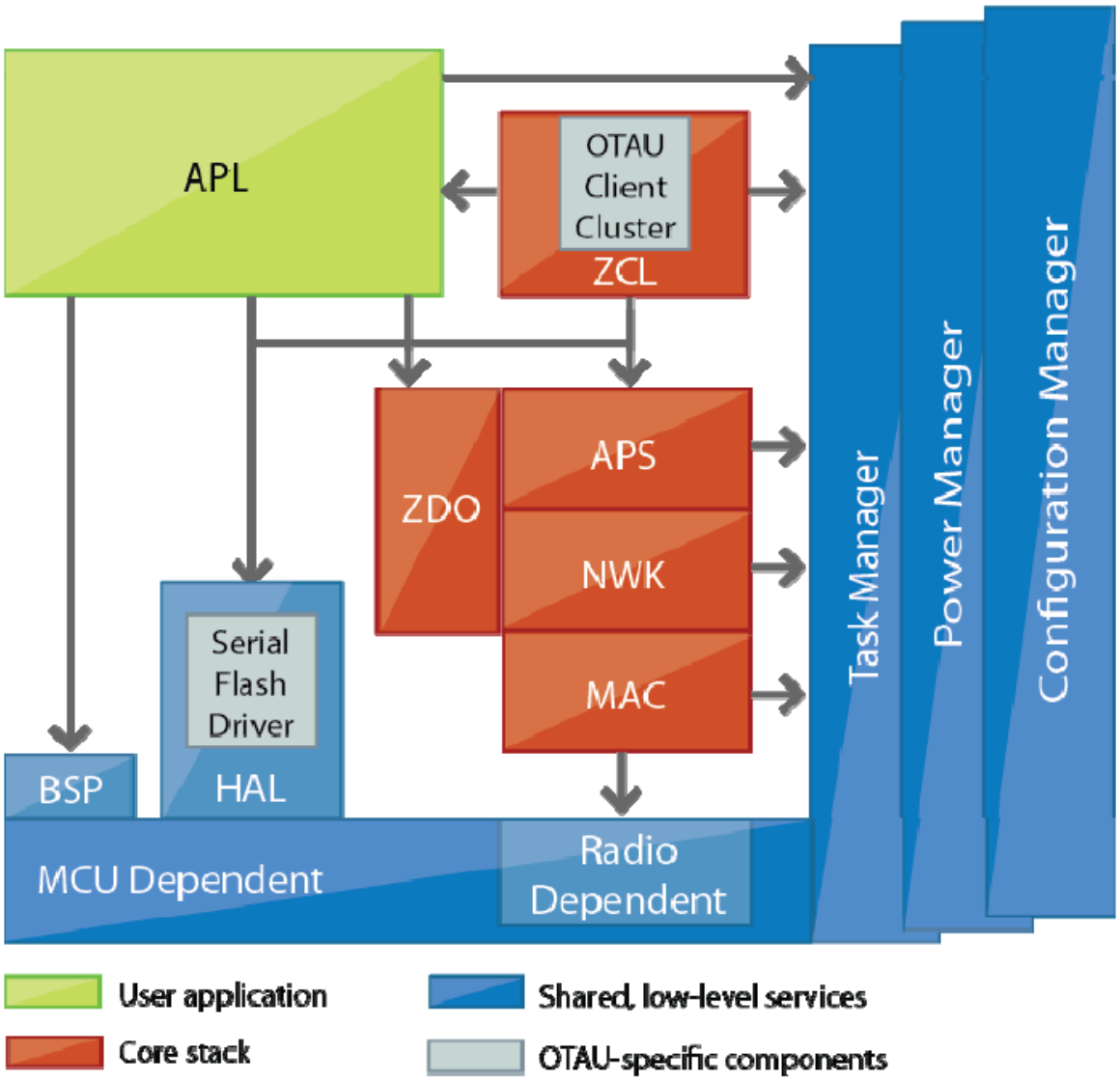


Figura 5.1. Arquitetura de um Cliente OTAU. [8]

Existe a possibilidade de se atualizar remotamente um dispositivo ZigBit estando este fora da rede. Para este caso, se diz que existe um ponto de acesso de atualização (Upgrade Access

Point – UAP) dedicado. Porém, para este caso, o princípio pelo qual surge a necessidade de se atualizar remotamente cada nó, que é a atualização de um nó com a rede funcionando, é ferido.

Aqui, parte-se do pressuposto de que existe uma rede de automação estabelecida e que o upgrade de cada nó se faz necessário com a rede funcionando. É o chamado ponto de acesso de atualização em rede (*in-network* UAP).

É possível de se programar um controlador da rede para ser um servidor OTAU, mas isso consome maior poder de processamento do mesmo. Caso seja uma rede muito grande, assim como acontece em redes de automação predial, isso pode diminuir a confiabilidade da rede, pois os deadlines de cada tarefa podem não ser cumpridos.

O passo-a-passo do protocolo para transferência de imagens de firmware é:

1. O servidor OTAU é ligado ao computador com o software Bootloader para o microcontrolador ATmega1281, aparecendo para os clientes OTAU (o cliente não faz parte da rede, apenas tem acesso ao canal de comunicação);
2. Cada cliente manda mensagens de confirmação para o servidor mostrando a ele que foi possível encontra-lo;
3. Se necessário, é estabelecida uma senha para ligação no sentido cliente-servidor;
4. A partir do Bootloader, a imagem a ser baixada para o cliente é registrada e fica pronta para ser enviada;
5. A imagem é enviada ao cliente;
6. É feita a verificação (*checksum*) para saber se a imagem foi recebida por completo;
7. O servidor envia um comando de ativação para o cliente (é enviado o comando de reset para o chip ATmega1281 do cliente, quando o chip é religado, a imagem é trocada);
8. O servidor recebe a confirmação de que a substituição de imagem de firmware foi realizada com sucesso;
9. O Bootloader avisa ao usuário que a atualização de firmware foi bem sucedida.

Uma grande vantagem oferecida pelo serviço de OTAU pela BitCloud é o funcionamento da rede e das funcionalidades de cada nó, inclusive o nó a ser atualizado, enquanto ocorre a atualização.

Uma grande desvantagem da utilização da aplicação OTAU é a necessidade de utilização de uma memória flash externa, para armazenamento do firmware recebido. A BitCloud assume que o firmware presente na memória interna do ZigBit é grande o suficiente, de forma que não se torna possível armazenar o novo firmware na memória interna, sendo necessária a presença de uma memória externa para tal função.

5.2 SOLUÇÃO PROPOSTA

Para habilitar a função OTAU, são necessárias algumas considerações no projeto:

- Modificações no Hardware;
- Construção e inclusão de bibliotecas OTAU;
- Definição de parâmetros OTAU;
- Utilização da API OTAU Cluster;
- Utilização correta do PC Tools.

5.2.1 MUDANÇAS NO HARDWARE

Mudanças no Hardware devem ser feitas. Deve-se levar em conta a plataforma e o pacote sustentados por esta aplicação. No caso do ZigBit, que possui um ATmega1281, é necessário que a placa tenha suporte para uma memória flash externa, que, neste caso, podem ser a Atmel AT25F2048 ou a AT45DB041. O tipo de memória externa deve ser definida também em nível de programação. A placa de desenvolvimento da Atmel, a Meshbean Board, possui suporte para estas memórias externas. A placa ZigBit Channel 1.0, desenvolvida para o fim deste Trabalho de Graduação, foi projetada sem esta funcionalidade. Portanto, não possui suporte para memória flash externa.

A biblioteca OTAU Cluster utiliza, no lado do servidor, a porta UART através do componente *Image Storage Driver* (ISD). É necessário, portanto, que o servidor tenha a porta UART dedicada somente a esta funcionalidade. No caso de se ter um servidor em um nó coordenador, deve-se tomar o devido cuidado para que a função de transferência de dados de rede não seja feita através da UART. Como toda a comunicação de módulo para módulo é feita através da porta UART, não se fez possível utilizar o nó coordenador como servidor OTAU. Utiliza-se, portanto, um módulo separado como servidor.

5.2.2 MUDANÇAS NO SOFTWARE

Devem ser feitas alterações no Software na maneira como se constrói a aplicação em nível de programação, alterando *makefiles*, adicionando bibliotecas desta funcionalidade e definindo funções e funcionalidades desta aplicação.

A maneira como se comporta cada nó na aplicação OTAU é designada a partir de definições cliente/servidor, assim como se deve definir canais de comunicação e endereçamento.

As bibliotecas da API ZCL, que define clusters (estruturas) geralmente bastante utilizados em diversos tipos de aplicações e funcionalidades, são utilizadas na aplicação OTAU. Isso significa que algumas de suas bibliotecas devem ser adicionadas na construção da imagem e declaradas no decorrer da programação. Também devem ser feitas as devidas modificações no *makefile* para que a imagem possa ser corretamente construída, incluindo o diretório dessas bibliotecas.

Dentro do header configuration.h, devem ser feitas as seguintes declarações/definições de parâmetros:

- Definir o tipo de memória externa utilizada tanto no servidor como nos clientes;
- Definir o número máximo de servidores na rede em cada cliente;
- Definir o número máximo de clientes a serem encontrados em cada servidor;
- Definir a duração entre duas tentativas falhas de se encontrar um servidor em cada cliente;
- Definir o endereço OTAU padrão em cada cliente.
- Definir qual o tipo do nó (cliente ou servidor) utilizando as definições de OTAU_CLIENT ou OTAU_SERVER;

Dentro do arquivo principal (EndDevice.c ou Coordenador.c), devem ser feitas as declarações de estruturas e de funções:

- Definir o tipo de estrutura do cluster ZCL_OtauCluster_t chamando ZCL_GetOtauClientCluster() para clientes ou ZCL_GetOtauServerCluster() para servidores;

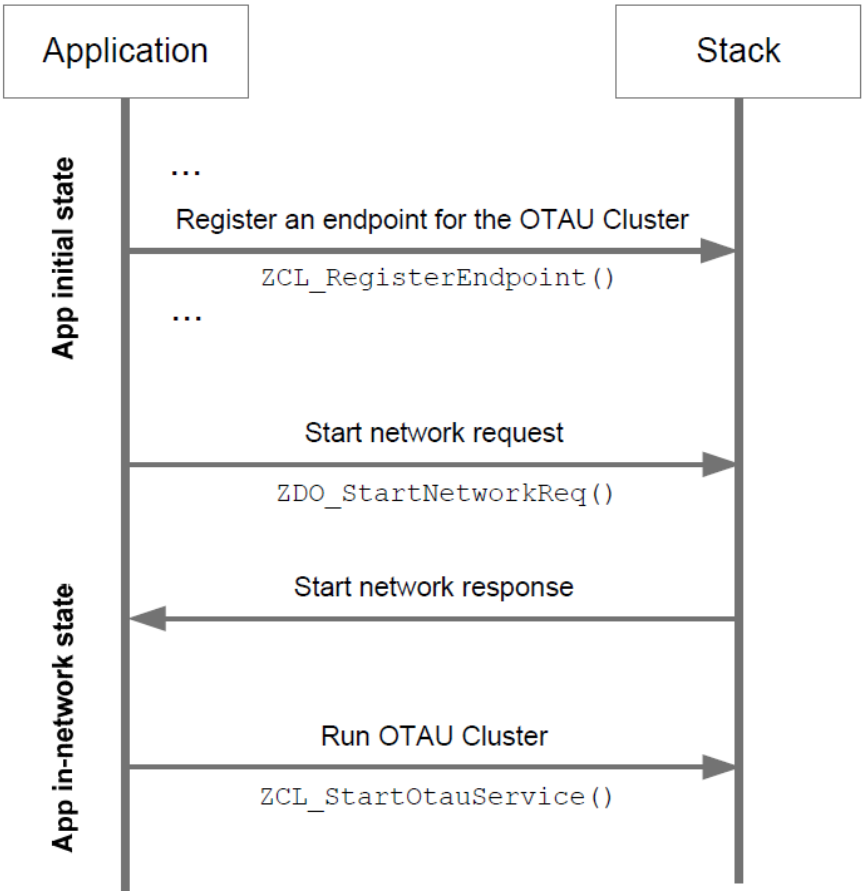


Figura 5.2. Interação para suporte do Cluster OTAU entre Aplicação e BitCloud. [8]

- Definir, no escopo, variáveis a serem utilizadas como parâmetros das funções API;
- Definir constantes que variam de acordo com o tipo de nó OTAU, como número de clusters de entrada e de saída;
- Receber o tipo de cluster OTAU;
- Registrar o ponto final de acesso, que é onde reside o serviço OTAU, ou seja, é o mapa para encontrar o acesso cliente/servidor. O registro do ponto final de acesso é feito através da função ZCL_RegisterEndpoint() (existem modificações de funcionamento desta de acordo com o tipo do nó OTAU);
- Inicializar o serviço OTAU. A inicialização do serviço OTAU é feita a partir da função ZCL_StartOtauService(&otauInitParams, otauClusterIndication) (existem modificações de funcionamento desta de acordo com o tipo do nó OTAU);
- Processar o serviço OTAU através da reinicialização do ATmega1281. Isso é feito a partir da função HAL_WarmReset().

5.3 IMPLEMENTAÇÃO

Conforme dito, a placa ZigBit Channel 1.0, projetada para fornecer os resultados nos experimentos do procedimento de Dormir, não foi projetada para fins de gravação remota. Isso porque não se sabia da necessidade de se ter uma memória externa para tal. Então, todos os testes foram feitos nas placas de desenvolvimento da Atmel, a Meshbean Board, que possui a entrada para a memória externa correta.

Sabe-se que é possível fazer todo o procedimento necessário para que a gravação remota funcione na ZigBit Channel 1.0, pois é o mesmo chip de controle e de comunicação de dados que existe na placa da Atmel. Porém, durante o envio do firmware de servidor para cliente, a imagem é descartada (não é armazenada), tornando-se impossível processar a substituição da mesma.

O que se faz então é a correta programação de cada um dos módulos em utilização. Foi montada uma rede ZigBee com um coordenador (conectado ao computador), um nó roteador com aplicação OTAU como cliente e um nó final, que é um subordinado sem funcionalidade OTAU. A rede é programada com aplicação WSN Demo e, ainda, com a funcionalidade de programação remota, conforme mostrado no Anexo V.

A aplicação WSN Demo é utilizada aliada a módulos MeshBean. Ela simplesmente disponibiliza, através do supervisório WSN Monitor, informações como parâmetros de rede (endereço do nó, canal de comunicação, etc) e também informações de monitoramento (temperatura, tensão da bateria, umidade) através dos sensores. O WSN Monitor mostra, também, o esquemático da rede, com coordenador e demais nós.

A funcionalidade OTAU não depende do tipo de aplicação utilizada, podendo ser aplicações *Peer to Peer*, WSN Demo, *Low Power*, dentre outros. A utilização da aplicação WSN Demo se encaixa bem neste experimento pela possibilidade de se visualizar a rede no software WSN Monitor, enxergando, assim, cada um dos nós da rede com seus respectivos parâmetros.

O servidor OTAU deve estar desligado enquanto não se deseja fazer nenhuma atualização de firmware em nenhum dos dispositivos da rede. Ele é programado de maneira separada, ou seja, é utilizado um módulo separado da rede para tornar-se o servidor. Quando conectado ao computador via serial, o servidor OTAU é ligado e passa a receber mensagens de confirmação de cada cliente OTAU da rede que se encontra em seu alcance.

É importante destacar que não é necessário ao servidor OTAU estar diretamente conectado ao nó que se deseja atualizar, pois os pacotes podem ser enviados pela rede de maneira indireta, passando de roteador a roteador (inclusive podem passar pelo coordenador da rede no meio do caminho) até chegar ao dispositivo que se deseja. Isso quer dizer que basta ao servidor estar no alcance de apenas um elemento da rede que, desta maneira, tem acesso a qualquer ponto da rede.

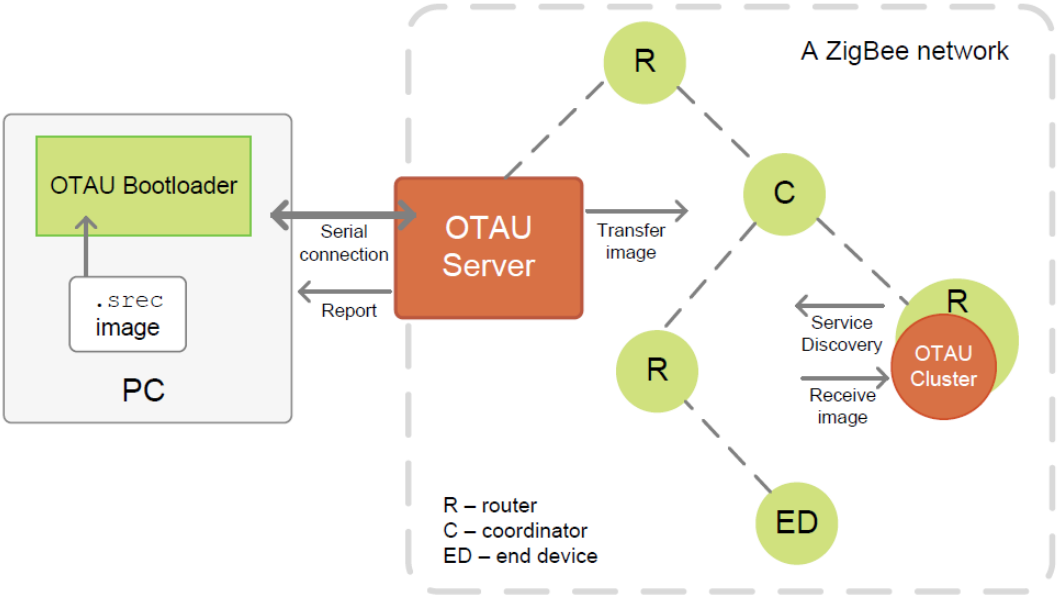


Figura 5.3. Gravação remota com aplicação OTAU. [8]

O servidor OTAU faz parte da rede ZigBee, sendo considerado um nó roteador. Ele possui acesso ao canal de comunicação. O teste pode ser feito utilizando a aplicação WSN Demo, juntamente com o supervisório WSN Monitor suportando a funcionalidade OTAU. É possível ao WSN Monitor encontrar cada um dos nós da rede (coordenador, roteadores e sensores), aparecendo, inclusive, o servidor OTAU, evidenciado como roteador. O endereço do servidor na rede deve ser diferente dos endereços de todos os demais nós.

Quando finalizado todo o processo de configuração das placas de desenvolvimento Meshbean, cabe ao usuário conectar o servidor ao PC Bootloader Tool via serial. Na aba OTAU (ou BitCloud OTAU, dependendo da versão) deste programa, é necessário alterar as configurações de rede, como *Channel Mask* e Network PanID, para que seja possível ao servidor encontrar a rede em questão.

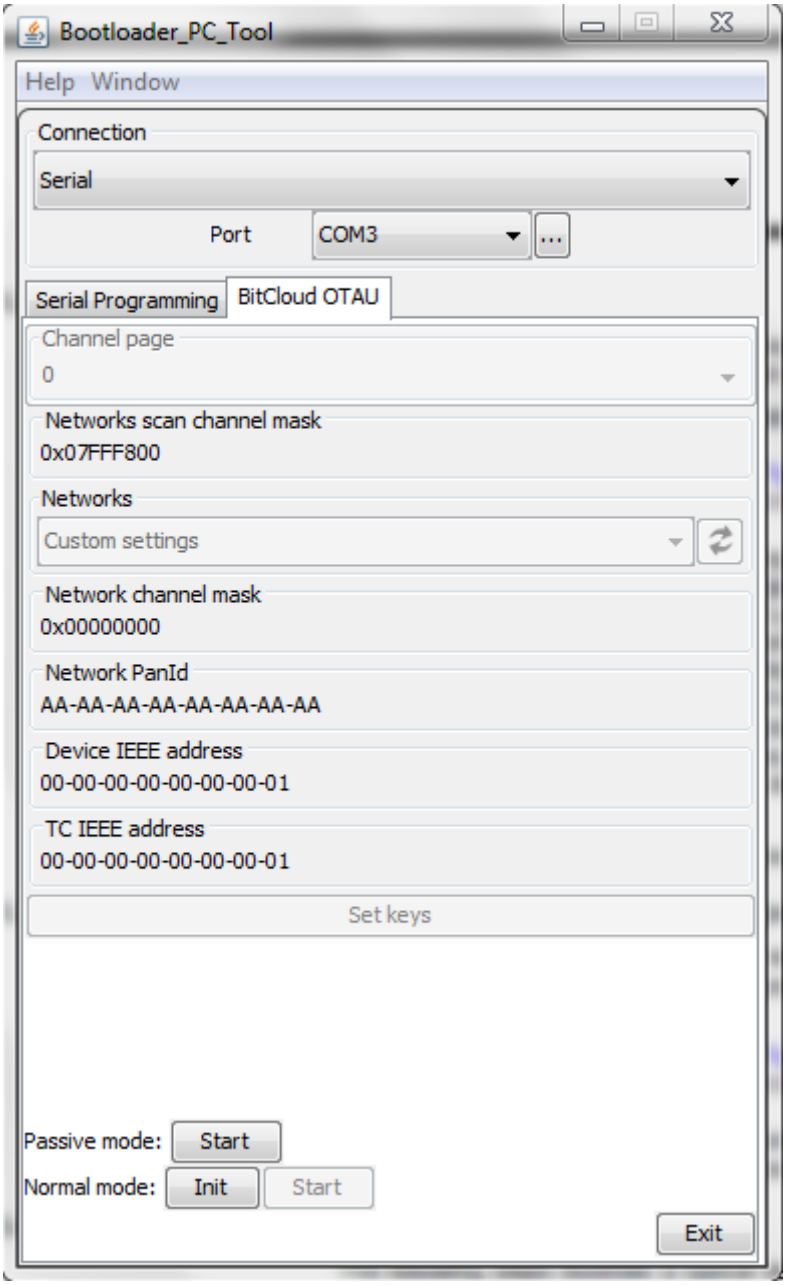


Figura 5.4. Aba BitCloud OTAU do software Bootloader PC Tool.

Inicia-se o processo com um clique em *Start Passive Mode*, que iniciará o processo de busca pelos dispositivos da rede com funcionalidade OTAU. Após alguns minutos, cada um dos dispositivos será mostrado em uma nova janela, sendo mostrada a opção de atualizar a imagem de cada um deles. Outras informações presentes nesta janela são o endereço do nó na rede, o fabricante do módulo, o tipo de imagem e qual a versão do firmware.

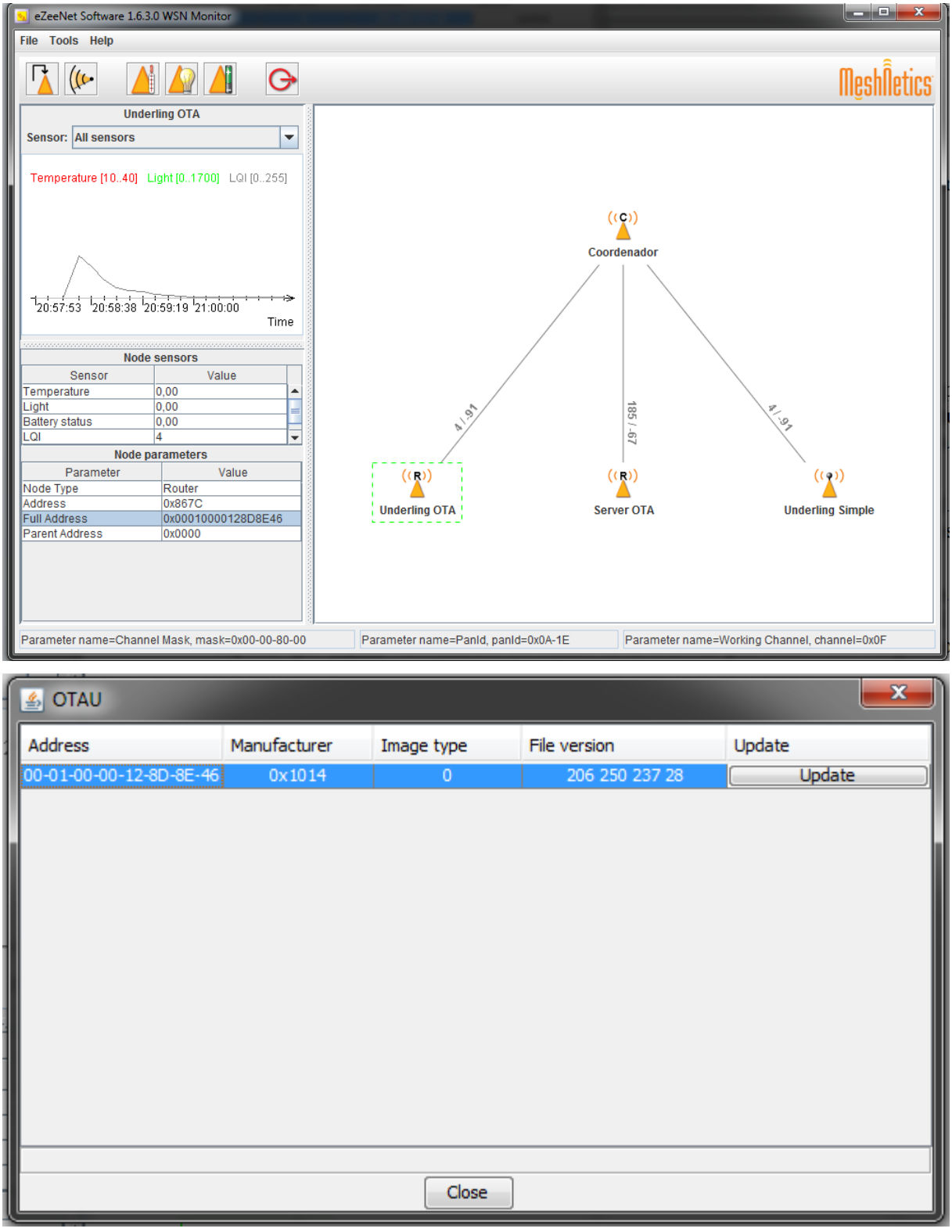


Figura 5.5. Janela de dispositivos encontrados pelo OTAU Bootloader.

Na versão do Bootloader (Bootloader_PC_Tool v1.2.2.214) que foi utilizado para os testes deste Trabalho, não é possível alterar os parâmetros de rede para uma busca pontual pela rede. Pelo contrário, é feita uma varredura dos canais que vão de 0x00000000 (canal zero) a 0x07FFF800, buscando por dispositivos de rede que se encontrem nestes canais.

Isto é uma boa ferramenta para a automação predial, pois se torna possível a um único servidor encontrar dispositivos de redes distintas, podendo ser atualizados vários dispositivos simultaneamente.

Por outro lado, a busca se torna muito mais lenta, uma vez que a varredura é bem mais complexa. Se o usuário deseja atualizar um dispositivo da rede 0x00000015 (canal vinte), deve esperar que sejam vasculhados os canais 0, 1, e assim por diante até chegar no vinte, o que pode demorar alguns minutos.

Ainda assim, mesmo depois de encontrado o dispositivo desejado, a transferência de pacotes, que ocorre juntamente com a verificação de suas chegadas, é bastante lenta. Esses tempos de resposta e de finalização de cada processo serão discutidos no próximo tópico.

5.4 RESULTADOS E ANÁLISE

Na rede montada, é possível ver pelo WSN Monitor que todos os dispositivos de rede estão conectados diretamente ao coordenador, pois todos estão em seu alcance. É possível ver que todos os dispositivos puderam ser encontrados, inclusive o Servidor OTAU, definido como roteador dentro da rede ZigBee.

Assim como mostrado na figura 4.26, a implementação da funcionalidade OTAU não compromete a aplicação WSN Demo. Nós com ou sem a funcionalidade são encontrados como deveria acontecer. Este princípio se estende para as demais aplicações. Uma aplicação *Peer to Peer*, por exemplo, deve funcionar perfeitamente com a adição da funcionalidade OTAU.

É possível enxergar também que, mesmo o servidor OTAU estando ligado ao coordenador e não diretamente ao cliente OTAU, a atualização se faz possível, provando o princípio de que basta ao servidor estar ligado a um nó da rede que se torna possível atualizar remotamente qualquer nó que se faz presente na mesma.

O processo de busca possui tempo de resposta que pode ser considerado aleatório. Muitos são os fatores que modificam o tempo de busca pelos dispositivos OTAU, como tempo de Dormir por dispositivo, distância do nó visualizada pelo servidor (deve passar por quantos dispositivos para alcançar o nó destino) e até mesmo nível de segurança da rede. Portanto, o tempo de busca pelos clientes OTAU não foi analisado.

A partir do momento em que os clientes são encontrados, passa a ocorrer o processo de envio de pacotes no sentido servidor/cliente. Este processo pode ser estimado, dependendo dos seguintes fatores:

- Tipo cliente OTAU: para clientes definidos como roteadores, o tempo de atualização é menor do que para dispositivos finais.
- Taxa de envios e recebimentos de confirmações (Polling Rate): quanto maior for a taxa de checagem dos dispositivos da rede ZigBee, mais rápida se torna o envio dos pacotes.
- Tipo de segurança da rede: para redes com baixa segurança (criptografia) de rede, ou até nenhuma, a atualização é mais rápida quando comparada com redes com alto nível de segurança.

O teste foi feito de acordo com a figura 4.26: um servidor OTAU declarado como roteador e um cliente OTAU declarado também como roteador. Para diferentes tamanhos de imagens baixadas, o tempo de gravação foi obtido e anotado. Os resultados podem ser vistos na Tabela 4.7 e através da Figura 4.27.

Tamanho	Tempo Gasto para Transmissão
92 kb	18m e 10s
105 kb	21m
122 kb	24m e 40s

. Tabela 5.1 –Dados de tempo do experimento de OTAU.

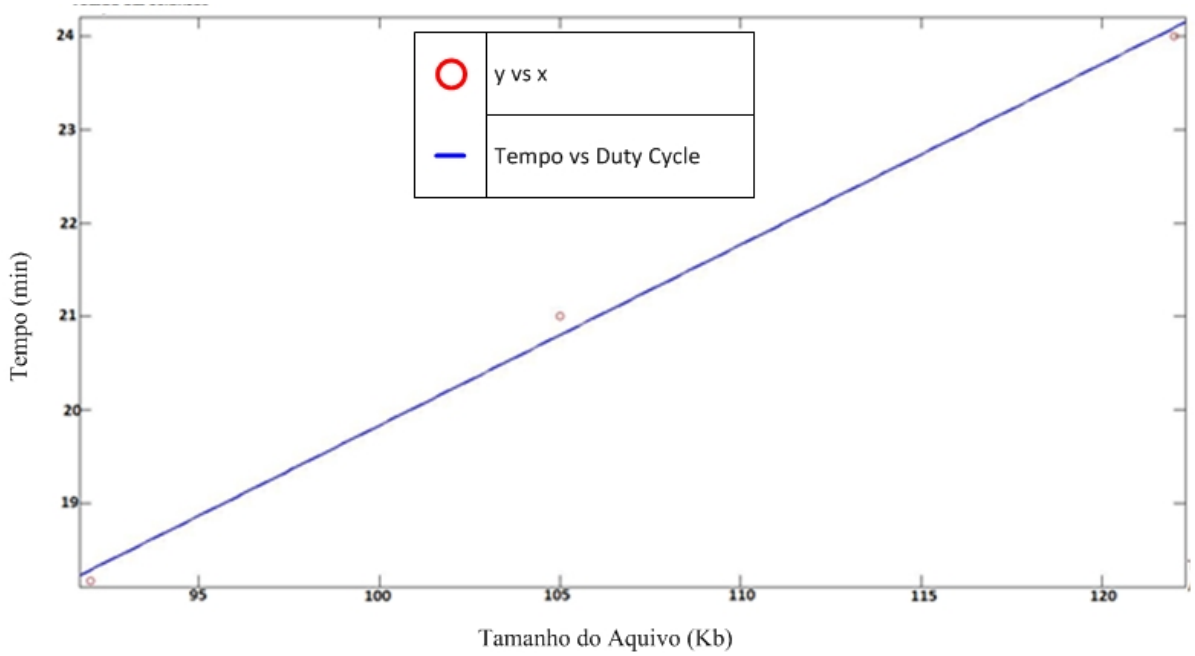


Figura 5.6. Gráfico Tempo vs. Tamanho do Arquivo.

É possível perceber que a curva de tempo de gravação pelo tamanho da imagem é bastante linear. O que já era esperado de uma rede robusta, com componentes que conseguem se comunicar sem a intrusão de muitos tipos de perturbações. Realmente se esperava que a proporção de tempo por tamanho do arquivo fosse constante.

Espera-se, para um cliente definido como Dispositivo-Final, que o tempo de resposta seja um pouco maior. Para o caso de uma rede que opere em uma taxa de comunicação maior, espera-se tempos de resposta menores.

A rede utilizada não possui nenhum tipo de criptografia inserida. Isso significa que para uma rede com alto nível de segurança, o tempo para gravação da imagem é ainda maior.

Depois de todos os pacotes serem enviados, e depois de já recebidos as mensagens de confirmação de envio e recebimento por parte dos dispositivos que fazem parte do processo de gravação remota, ocorre o Warm Reset. Esta é uma mensagem de comando de servidor para cliente avisando ao cliente que ele deve ser reiniciado para que a nova imagem possa ser aplicada.

Neste caso, a memória flash externa não se encontra presente. Isso significa que todos os pacotes recebidos pelo cliente foram descartados (foram enviados para o pinos onde devia estar conectada a memória externa). Assim, quando ocorre o Warm Reset no ATmega1281, não há reconhecimento de novo firmware em lugar algum da placa. Não ocorre, portanto, a finalização da gravação remota.

Mesmo que a gravação não possa ser concluída, é possível perceber, através do Bootloader PC Tool e do WSN Monitor, que este procedimento deve funcionar para um hardware com a presença da memória flash externa.

O Bootloader PC Tool mostra ao usuário todas as etapas de envio da nova imagem, provando que todos os pacotes são devidamente recebidos em seus destinos.

Através do WSN Monitor é possível ver que o nó cliente continua na rede durante a atualização trabalhando de acordo com as rotinas pré-estabelecidas a ele via software e que ao fim do processo de envio e consequente *reset* do módulo, o nó desaparece da rede, voltando logo em seguida. Isso porque sofreu processo de desligamento para atualização de imagem, como esperado.

6 CONCLUSÃO

A partir dos experimentos e análises feitas ao longo deste trabalho, constataram-se importantes fatos a respeito dos problemas analisados sumariamente.

Em relação ao gasto de energia, pode-se visualizar que o tratamento deve ser feito a todos os níveis de implementação, tanto hardware como software. Os protocolos implementados tiveram, embora os resultados finais tenham sido aquém dos esperados, um comportamento satisfatório. Fica claro que a implementação de hardware dificultou a melhor análise dos protocolos de software, mas ainda assim, fica claro também a eficiência destes protocolos. Quanto mais utilizado o protocolo de economia de energia, mais o gasto de energia foi reduzido, possibilitado a conclusão óbvia de que o protocolo de Dormir funciona. Além disso, a partir dos dados coletados fica claro também que a utilização de protocolos de economia de energia é fundamental para se utilizar a rede em seu estado completamente maleável, ou seja, sem fiação.

A economia de energia na rede de comunicação a partir da utilização do procedimento de Dormir pode proporcionar, portanto, melhorias significativas em uma rede de automação predial. Isso porque diminui consideravelmente o consumo de energia, como foi mostrado ao longo do capítulo 4, diminuindo custos com operação predial. Além disso, pelo aumento da vida útil de baterias, diminui custos de manutenção, tanto no sentido monetário com no sentido temporal. Baterias que duram mais necessitam ser trocadas com menor frequência, aumentando a confiabilidade da rede.

Analisando o outro ponto abordado no trabalho, a gravação remota, que facilitaria a programação dos nós da rede, fazendo com que o trabalho ficasse mais confortável, obteve-se dados e resultados limitados. Em face de o hardware utilizado não proporcionar condições para a utilização desta vantagem, somente pudemos simular a sua funcionalidade. É fato de que com esta função o trabalho com a rede fica extremamente simplificado, mas chama-se a atenção para o fato de o consumo de tempo para a utilização deste protocolo ser razoavelmente grande como pudemos ver no capítulo 4.

A facilidade da programação remota se faz, portanto, necessária a qualquer tipo de rede sem fio de automação predial. O aumento no número de nós por rede aumenta a complexidade da mesma, dificultando cada vez mais o diagnóstico de problemas de rede e sua consequente resolução através de métodos mais simples. A programação remota é, portanto, juntamente com um sistema supervisório de pontos e dispositivos de rede, a ferramenta certa para diagnóstico e resolução de problemas de maneira remota, sem a necessidade de operação em campo.

Ao final do trabalho, pode-se dizer que os resultados, embora um pouco aquém do esperado, foram atingidos de maneira razoavelmente satisfatória. protocolos de redução de gasto de energia tiveram sua funcionalidade comprovada e a programação e plataforma para a utilização da programação remota foi feita e o terreno preparado para os próximos trabalhos.

7 PROPOSTAS PARA TRABALHOS FUTUROS

7.1 MODIFICAÇÕES DE HARDWARE

7.1.1 MODIFICAÇÕES NA ALIMENTAÇÃO

Para a melhor economia de energia, como visualizamos no capítulo 4 o ideal é que se retire o regulador de tensão dos nós Dispositivo-Final que funcionam com bateria e seja utilizada uma bateria na faixa de funcionamento direta do ZigBit. Desta maneira, o dispositivo ZigBit acoplado na placa ZigBit Channel 1.0 não terá o gasto do regulador de tensão e consequentemente terá um melhor aproveitamento da carga útil da bateria.

7.1.2 MEMÓRIA FLASH EXTERNA

Para a utilização da vantagem que o ZigBit e Bitcloud oferecem, de gravação remota, utiliza-se uma memória flash referenciada na programação. no entanto, nem as placas da MeshBean possuem tal memória, então se faz imperativo a utilização de uma conexão de memória flash externa. para a associação de memória externa ao ZigBit, as ligações devem ser feitas da seguinte maneira

- USART0_RXD(PE0) -> MISO (external flash)
- USART0_TXD(PE1) -> MOSI (external flash)
- USART0_EXTCLK(PE2) -> SCK (external flash)
- PF3 -> CS (external flash) as defined in ofdMemoryDriver.h (HAL_ASSIGN_PIN(EXT_MEM_CS, F, 3)

Com essas ligações feitas, o dispositivo estaria apto para utilizar de maneira completa a gravação remota.

7.2 MODIFICAÇÕES DE SOFTWARE

A estrutura de programação deve ser diferenciada. Nos experimentos realizados neste trabalho não houve atuadores, por isso a programação ficou mais direcionada para a troca de dados exclusivamente. Com um nó atuador na rede, e em face da troca de informações com um supervisório o algoritmo da programação do coordenador mudaria, como mostrado no item 1 Anexo VII.

Note que o coordenador agora tem também a tarefa de traduzir o protocolo ModBus para conversar com o supervisório.

O algoritmo da programação do atuador seria como visto no item 2 do Anexo VII.

Note que a programação do atuador seria extremamente semelhante à programação do Dispositivo-Final vista na Figura 4.21, a qual permanece a mesma. A única diferença é que por estar conectado á rede elétrica, o atuador não entra em modo Dormir.

REFERÊNCIAS

[1] Gislason, D. **ZIGBEE WIRELESS NETWORKING**. Newness, 2008.

[2] Queiroz, RB., Azevedo, R.C.A. **REDE DE SENSORES SEM FIO PARA AUTOMAÇÃO PREDIAL COM MÓDULOS MESHBEAN.** Trabalho de Graduação. Universidade de Brasília. 2009.

[3] Evangelista, D.S. **INTEGRAÇÃO DE REDES DE SENSORES ZIGBEE PARA AUTOMAÇÃO PREDIAL UTILIZANDO MÓDULOS MESHBEAN.** Trabalho de Graduação. Universidade de Brasília. 2010.

[4] Junior, J.U.D., Novais, M.C.C. **INSTRUMENTAÇÃO E CONTROLE DE UM SISTEMA DE AR CONDICIONADO HÍBRIDO UTILIZANDO BACNET SOBRE ZIGBEE.** Trabalho de Graduação. Universidade de Brasília. 2010.

[5] MeshNetics. **ZIGBIT DEVELOPMENT KIT 1.3 USER'S GUIDE.** Doc. S-ZDK-451~01 v1.10. MesNetics. Dezembro de 2007.

[6] Atmel. **DATASHEET ZIGBIT 2.4GHZ WIRELESS MODULES ATZB-24-A2/B0.** Atmel, 2009.

[7] Atmel. **ATMEL BITCLOUD: Developer Guide.** Rev 8199G-MCU Wireless-11/11. Atmel AVR250. 2011.

[8] Atmel. **BITCLOUD OTAU USER GUIDE.** Rev 8426B-AVR-11/11. Atmel AVR258. 2011.

[9] Atmel. **BITCLOUD STACK API REFERENCE.** V1.13.0. Atmel. 2011.

[10] Atmel. **SERIAL BOOTLOADER USER GUIDE.** Rev 8390B-AVR-11/11. Atmel AVR2054. 2011.

[11] Departamento de Serviços – Núcleo de Estudos. **SISTEMA DE GERENCIAMENTO PREDIAL: Treinamento Módulo 6.** Ver 18. Johnson Controls. 2012.

[12] **EVALUATING THE DIFFERENT PROTOCOLS FOR LOW POWER WIRELESS NETWORKS IN INDUSTRIAL AUTOMATION**.Iniciativa: Publitek European Editors. Disponível em: http://www.digikey.com/us/en/techzone/wireless/resources/articles/evaluating-the-different-protocols.html. 2012

[13] **DURACELL COPPERTOP DATASHEET – Alkaline-Manganese Dioxide Battery.** MN1604. Duracell. Disponível em: http://www.duracell.com/media/en-US/pdf/gtcl/Product_Data_Sheet/NA_DATASHEETS/MN1604_US_CT.pdf

[14] **HOME AND BUILDING AUTOMATION.** Research & Development. Spintel. Disponível em: http://www.spintel.it/SPINTEL/index.php?option=com_content&view=article&id=19&Itemid=2&lang=en

[15] **ZIGBEE ALLIANCE.** Disponível em: http://www.zigbee.org/

[16] **IEEE 802.15.4.** Disponível em: http://www.802..org/. 2012.

[17] **REDES DE SENSORES SEM FIO.** Disponível em: http://www.vivasemfio.com/blog/tag/zigbee/. Maio de 2009.

[18] Messias, A. R., **CONTROLE REMOTO E AQUISIÇÃO DE DADOS VIA XBEE/ZIGBEE (IEEE 802.15.4).** Disponível em: http://www.rogercom.com/ZigBee/ZigBee.htm. 2008.

ANEXOS

ANEXO I

Código Comentado do Módulo Coordenador

```
/*
 * Coordenador.c
 * Programa que funciona como um coordenador ZigBit para uma rede básica. As funções do coordenador são iniciar a rede, receber as medições do EndDevice e enviá-las ao PC.
 * Created: 22/01/2013 21:05:08
 *  Author: Lucas Guilhem de Matos - lucas.rato@gmail.com
 *       Para Mais informações a respeito das funções e do funcionamento da rede ZigBee, recomenda-se a leitura da API da BitCloud.
 */

/******************************************************************************
***********************************************
                Nota, incluir também as bibliotecas exclusivas da aplicação e os .Cs que sejam customizados
                É importante ao compilar o programa verificar se existe o caminho referenciado para o include das bibliotecas.
******************************************************************************
***********************************************/
#include <types.h>
#include <taskManager.h>
#include <configServer.h>
#include <zdo.h>
#include <peer2peer.h>
#include <serialInterface.h>
#include <appTimer.h>
#include <adc.h>

/******************************************************************************
***********************************************
                Definição de variáveis globais
******************************************************************************
***********************************************/
static AppState_t appState = APP_INITIAL_STATE;
                                //Estado da aplicação. Usado como referência no escalonador de tarefas da aplicação.
static ZDO_StartNetworkReq_t networkParams;
                                //Parâmetros da rede que será gerada/acessada
static SimpleDescriptor_t simpleDescriptor = { APP_ENDPOINT, APP_PROFILE_ID, 1, 1, 0, 0 , NULL, 0, NULL };   //Simple Descriptor. Parâmetros do Nó
static APS_RegisterEndpointReq_t endpointParams;
                        //Parâmetros para registro do nó na rede.

static HAL_UsartDescriptor_t appUsartDescriptor;
                //Parâmetros para a configuração da porta Serial
static uint8_t usartRxBuffer[100];
                                //Buffer de recepção da porta serial
```

```
static uint16_t nwkAddr;
                                        // Receberá o Short_Adress do nó
static AppMessageBuffer_t appMessageBuffer;
                                    // Buffer utilizado para o envio de Daos Wireless.

/************************************************************************
*************************************************
        Function Prototype Section
************************************************************************
*************************************************/
static void initNetwork(void);
                                        // Função que inicializa os parâmetros de rede e define o
papel do nó na rede que se fromará/acessará
static void startNetwork(void);
                                        //      Inicializa/Acessa      a      rede.static      void
APS_DataIndication(APS_DataInd_t* dataInd);  //Indicação de dao recebido.
static void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo);
        //Confirmação de formação de rede. Callback

static void initSerialInterface(void);
                        //Define os parâmetros de comunicação serial
static void usartBytesReceived(uint16_t readBytesLen);
                //Recebimento de dados via serial

static void APS_DataIndication(APS_DataInd_t* indData);
                        //Função que trata o recebimento de dados.

void ZOD_WakeUpInd(void);
                                            //Função de Stub Para o Coordenador.

/************************************************************************
*************************************************
        Stub Functions
************************************************************************
*************************************************/
void ZDO_MgmtNwkUpdateNotf(ZDO_MgmtNwkUpdateNotf_t *nwkParams)
{
        (void)nwkParams;
}


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

void ZDO_UnbindIndication(ZDO_UnbindInd_t *unbindInd)
{
  (void)unbindInd;
}
```

```
static void usartWriteConf(void)
{
}

void ZDO_WakeUpInd(void)
{
}


/************************************************************************
*************************************************
                Rotinas para inicialização de rede.
************************************************************************
*************************************************/
//Inicializa a rotina de preparação da rede
static void initNetwork(void)
{
 CS_WriteParameter(CS_NWK_UNIQUE_ADDR_ID, &(bool){true});

 DeviceType_t deviceType;   //Papel do nó na rede.

 // Short Address
 CS_ReadParameter(CS_NWK_ADDR_ID, &nwkAddr);

 // Função do nó. Se for 0, o nó é coordenador, senão, é EndDevice.
 if (0 == nwkAddr)
 {
#ifdef _SECURITY_
   {
    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_END_DEVICE;
 }
 // Define o tipo do nó
 CS_WriteParameter(CS_DEVICE_TYPE_ID, &deviceType);

 // Muda o estado para a referência do escalonador de tarefas,
 appState = APP_NETWORK_JOINING_STATE;
}
```

```
//Quando um pedido de inicialização de rede é feito e a função é executada, então essa função
é chamada para exercer confirmação
void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo)
{
 if (confirmInfo->status == ZDO_SUCCESS_STATUS)
 {
        BSP_OnLed(LED_RED);
        appState = APP_NETWORK_JOINED_STATE;
  // Configura o nó a ser registrado
  endpointParams.simpleDescriptor = &simpleDescriptor;
  endpointParams.APS_DataInd = APS_DataIndication;
  // Register endpoint
  APS_RegisterEndpointReq(&endpointParams);
        SYS_PostTask(APL_TASK_ID);
 }
 else
 {
        //Volta ao escalonador de tarefas para continuar o processo.
  SYS_PostTask(APL_TASK_ID);
 }
}

//Função que inicia a rede.
static void startNetwork(void)
{
        //Determina a função de confirmação de envio.
        networkParams.ZDO_StartNetworkConf = ZDO_StartNetworkConf;
        //Começa a Rede
        ZDO_StartNetworkReq(&networkParams);
}


/**************************************************************************
**************************************************
                Configuração de Comunicação via USART/UART
**************************************************************************
**************************************************/

//Parâmetos para início de interface serial
static void initSerialInterface(void)
{
 appUsartDescriptor.tty            = USART_CHANNEL;
        //DEfine a maneira de transmissao
 appUsartDescriptor.mode           = USART_MODE_ASYNC;
        //Modo de transmissão. Sugestão para o modo assíncrono.
 appUsartDescriptor.baudrate       = USART_BAUDRATE_38400;
        //BaudRate.
 appUsartDescriptor.dataLength     = USART_DATA8;
        //Tamanho do Elemento de transmissão.
```

```
  appUsartDescriptor.parity          = USART_PARITY_NONE;
        //Paridade
  appUsartDescriptor.stopbits        = USART_STOPBIT_1;
        //Bits de Parada
  appUsartDescriptor.rxBuffer        = usartRxBuffer;
        //Buffer de REcepção
  appUsartDescriptor.rxBufferLength  = sizeof(usartRxBuffer);
        //Tamanho do Buffer de Recepção
  appUsartDescriptor.txBuffer        = NULL;
                // Callback mode. Quando a transmissão acaba... chama uma função de
confirmação
  appUsartDescriptor.txBufferLength  = 0;
  appUsartDescriptor.rxCallback      = usartBytesReceived;
        //Callback de recepção de mensagens
  appUsartDescriptor.txCallback      = usartWriteConf;
        //Confirmação de envio de dados
  appUsartDescriptor.flowControl     = USART_FLOW_CONTROL_HARDWARE;
                //Controle de fluxo.

  HAL_OpenUsart(&appUsartDescriptor);
}

//Essa função é só um eco para o teste. no funcionamento real da rede, é só uma função de
//Stub
static void usartBytesReceived(uint16_t readBytesLen)
{
        READ_USART(&appUsartDescriptor,                    appMessageBuffer.data,
APP_MAX_PACKET_SIZE);
        WRITE_USART(&appUsartDescriptor,(uint8_t*)&appMessageBuffer.data,
sizeof(appMessageBuffer.data));
        (void)readBytesLen;
}

/************************************************************************************
*************************************************
                Configurar parâmetros de troca de mensagens por rede
************************************************************************************
*************************************************/

// Todo dado recebido é escrito na porta serial
static void APS_DataIndication(APS_DataInd_t* indData)
{
        WRITE_USART(&appUsartDescriptor,indData->asdu,indData->asduLength);
}

/************************************************************************************
*************************************************
                Task Handler Function
************************************************************************************
*************************************************/
```

```
void APL_TaskHandler(void)
{
 switch (appState)
 {
  case APP_INITIAL_STATE:                          // nó inicial do nó
        initSerialInterface();
    initNetwork();
    SYS_PostTask(APL_TASK_ID);                     // Executa o próximo passo.
        break;

  case APP_NETWORK_JOINING_STATE:
        startNetwork();
        break;

  case APP_NETWORK_LEAVING_STATE:
            break;

      case APP_NETWORK_JOINED_STATE:
            break;
      default:
            break;
 }
}

/*************************************************************************
*************************************************
            Main Function
*************************************************************************
*************************************************/
//Inicializa o Micro-Controlador e chama a tarefa.
int main(void)
{
 SYS_SysInit();

      for(;;)
 {
  SYS_RunTask();
 }
}
```

ANEXO II

Código Comentado do Módulo Dispositivo-Final

```
/*
 * EndDevice.c
 * Programa que funciona como um dos nós da  rede implementada. A função deste nó é simplesmente entara num ciclo de Sleep/Temporização e enviar os dados medidos no ADC
 * Created: 22/01/2013 21:05:08
 *  Author: Lucas Guilhem de Matos - lucas.rato@gmail.com
 *       Para Mais informações a respeito das funções e do funcionamento da rede ZigBee, recomenda-se a leitura da API da BitCloud.
 */

/************************************************************************************************************************/
                Include Section. Nota, incluir também as bibliotecas exclusivas da aplicação, .Cs que sejam customizados
                É importante ao compilar o programa verificar se existe o caminho referenciado para o include das bibliotecas.
************************************************************************************************************************/
#include <types.h>
#include <taskManager.h>
#include <configServer.h>
#include <zdo.h>
#include <peer2peer.h>
#include <serialInterface.h>
#include <appTimer.h>
#include <adc.h>

/************************************************************************************************************************/
                Definição de variáveis globais
************************************************************************************************************************/
static AppState_t appState = APP_INITIAL_STATE;
                        //Estado da aplicação. Usado como referência no escalonador de tarefas da aplicação.
static ZDO_StartNetworkReq_t networkParams;
                        //Parâmetros da rede que será gerada/acessada
static SimpleDescriptor_t simpleDescriptor = { APP_ENDPOINT, APP_PROFILE_ID, 1, 1, 0, 0 , NULL, 0, NULL };   //Simple Descriptor. Parâmetros do Nó
static APS_RegisterEndpointReq_t endpointParams;
                //Parâmetros para registro do nó na rede.

static HAL_AppTimer_t delayTimer;
                                //Parâmetros para configuração do Timer
static uint8_t msgBuffer[] = "Sensor4";
                // Mensagem teste do programa.
```

```
static uint16_t nwkAddr;
                                // Receberá o Short_Adress do nó
static AppMessageBuffer_t appMessageBuffer;
                        // Application message buffer
static APS_DataReq_t apsDataReq;
                        // Parâmetros de Configuraçõ de Envio de dado na rede
static int i = 0;
                                // Variável inteira para Utilização geral
static ZDO_SleepReq_t zdoSleepReq;
                                //Parâmetros para pedido de Sleep.
static uint8_t adcBuffer;
                                //Buffer onde os dados do ADC são colocados
static HAL_AdcParams_t adcParams;
                                //Parâmetros para utilização do ADC
static int contdata;
                                //Variável para reforço no enviao de mensagem.
static ZDO_ZdpReq_t leaveReq;
                                //Parâmetros para deixar a rede.
/************************************************************************
*************************************************
                Function Prototype Section
************************************************************************
*************************************************/

///FUNÇÕES DE INICIALIZAÇÃO DE REDE

static void initNetwork(void);
                        // Função que inicializa os parâmetros de rede e define o papel do nó na
rede que se fromará/acessará
static void startNetwork(void);
                        //          Inicializa/Acessa          a          rede.static          void
APS_DataIndication(APS_DataInd_t* dataInd);  //Indicação de dao recebido.
static        void        ZDO_StartNetworkConf(ZDO_StartNetworkConf_t*        confirmInfo);
        //Confirmação de formação de rede. Callback
static void leaveNetwork(void);
                        //Função que prepara a saída de rede.
static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp);
        //Resposta à tentativa de deixar a rede.

///FUNÇÕES DE COMUNICAÇÃO SERIAL

static void usartBytesReceived(uint16_t readBytesLen);
        //Recebe os dados da serial. No caso do EndDevice, que não está conectado ao PC, é
uma Stub Function

///FUNÇÕES DE TEMPORIZAÇÃO

static void StartTimer(void);
                //Inicializa o Timer.
```

```
///FUNÇÕES DE COMUNICAÇÃO WIRELESS

static void networkSendData(void);
                //Função que prepara as configurações e envia um dado na rede.
static void APS_DataConf(APS_DataConf_t* confInfo);
        //Função que verifica se o dado foi enviado corretamente.
void APS_DataIndication(void);
                        //Função de interrupção que é chamada quando um dado é recebido.

///FUNÇÕES DE PROTOCOLO DE ECONOMIA DE ENERGIA

static void StartSleep(void);
                //Função que inicia o período de Sleep
static void ZDO_SleepConf(ZDO_SleepConf_t *conf);
        //Função que verifica se o Sleep foi iniciado com sucesso.

static void adcGetValue(void);
                        //Função que Habilita e lê o dado do ADC
static void initAdcParams(void);
                //Configura os parâmetros a serem usados pelo ADC
static void adcMeasured(void);
                        //Confirmação de que a medição do ADC foi realizada.

/**************************************************************************
**************************************************
                Stub Functions
                Nenhuma dessas funções faz nada, mas precisam ser declaradas, pois são
funções canônicas da pilha.
**************************************************************************
**************************************************/
void ZDO_MgmtNwkUpdateNotf(ZDO_MgmtNwkUpdateNotf_t *nwkParams)
{
        (void)nwkParams;
}


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

void ZDO_UnbindIndication(ZDO_UnbindInd_t *unbindInd)
{
  (void)unbindInd;
}

/**************************************************************************
**************************************************
                Rotinas para inicialização de rede.
```

```
**********************************************************************
**********************************************/
//Esta função inicializa os parâmteros e definições do nó apra que o mesmo se junte à rede.
static void initNetwork(void)
{
  CS_WriteParameter(CS_NWK_UNIQUE_ADDR_ID, &(bool){true});

  DeviceType_t deviceType;   //Papel do nó na rede.

  // Leitura do Short Adress do Nó
  CS_ReadParameter(CS_NWK_ADDR_ID, &nwkAddr);

  // Se o nwkAddr for 0 então o nó trabalhará como um coordenador
  // Se for outro valor, então o nó é um EndDevice.
  if (0 == nwkAddr)
  {
#ifdef _SECURITY_
    { //Coloca o UID do coordenador igual a um valor específico e seguro
      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_END_DEVICE;
          #define CS_RX_ON_WHEN_IDLE false
  }
  // Seleciona o tipo do Dispositivo
  CS_WriteParameter(CS_DEVICE_TYPE_ID, &deviceType);

  // Muda o estado. a partir de agora o nó Inicia ou se junta à rede com os parâmetros
definidos.
  appState = APP_NETWORK_JOINING_STATE;
}

//Quando um pedido de inicialização de rede é feito e a função é executada, então essa função
é chamada para exercer confirmação
void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t* confirmInfo)
{
  if (confirmInfo->status == ZDO_SUCCESS_STATUS)
  {
          appState = APP_NETWORK_JOINED_STATE;
          // Configura o EndPoint da aplicação.
    endpointParams.simpleDescriptor = &simpleDescriptor;
    endpointParams.APS_DataInd = APS_DataIndication;
    // Esta Função registra o EndPoint na rede.
```

```
   APS_RegisterEndpointReq(&endpointParams);
        SYS_PostTask(APL_TASK_ID);
  }
  else
  {
        //Chama novamente a tarefa para a proxima etapa.
   SYS_PostTask(APL_TASK_ID);
  }
}

//Função que inicia a rede.
static void startNetwork(void)
{
        //Confirmação de inicialização de rede.
        networkParams.ZDO_StartNetworkConf = ZDO_StartNetworkConf;
        // Função que inicializa a rede.
        ZDO_StartNetworkReq(&networkParams);
}

//Parâmetros para saída de rede.
static void leaveNetwork(void)
{
 ZDO_MgmtLeaveReq_t *zdpLeaveReq = &leaveReq.req.reqPayload.mgmtLeaveReq;
 APS_UnregisterEndpointReq_t unregEndpoint;

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

 leaveReq.ZDO_ZdpResp =  zdpLeaveResp;
 leaveReq.reqCluster = 1;
 leaveReq.dstAddrMode = SHORT_ADDR_MODE;
 leaveReq.dstNwkAddr = 0;
 zdpLeaveReq->deviceAddr = 0;
 zdpLeaveReq->rejoin = 1;
 zdpLeaveReq->removeChildren = 1;
 zdpLeaveReq->reserved = 0;
 ZDO_ZdpReq(&leaveReq);


 //Tenta se reunir à rede.
 appState = APP_NETWORK_JOINING_STATE;
 SYS_PostTask(APL_TASK_ID);
 }

static void zdpLeaveResp(ZDO_ZdpResp_t *zdpResp)
{
 //Stub Function
 (void)zdpResp;
}
```

```
/*********************************************************************************************
*********************************************
                Função para Inicialização e Uso do Temporizador
*********************************************************************************************
*********************************************/
static void StartTimer(void)
{
        delayTimer.interval = 1000L;
        delayTimer.mode     = TIMER_ONE_SHOT_MODE;
        delayTimer.callback = StartSleep;
        HAL_StartAppTimer(&delayTimer);
}

/*********************************************************************************************
*********************************************
                Configurar parâmetros de troca de mensagens por rede
*********************************************************************************************
*********************************************/
static void networkSendData(void)
{

         //for(i=0; i<sizeof(msgBuffer); i++)
         //appMessageBuffer.data[i] = msgBuffer[i];
         //i=0;
         apsDataReq.dstAddrMode = APS_SHORT_ADDRESS;              // Modo de envio
baseado no Shot Address
    apsDataReq.dstAddress.shortAddress = 0x0000;                          //Endereço
do lugar para qual ser enviado.
    apsDataReq.profileId = simpleDescriptor.AppProfileId;     // Profile ID
    apsDataReq.dstEndpoint = simpleDescriptor.endpoint;       // EndPoit de destino
    apsDataReq.clusterId = APP_CLUSTER_ID;                   // ID do cluster de destino
    apsDataReq.srcEndpoint = simpleDescriptor.endpoint;       // EndPoint da fonte
    apsDataReq.asdu = (uint8_t*) &appMessageBuffer.data;       // Ponteiro para o buffer de
aplicação
    // actual application message length
    //apsDataReq.asduLength = sizeof(msgBuffer)+1;
    apsDataReq.txOptions.acknowledgedTransmission  =  1;              // Transmissão com
reconhecimento ativada.
#if APP_FRAGMENTATION
    apsDataReq.txOptions.fragmentationPermitted = 1;
#else
    apsDataReq.txOptions.fragmentationPermitted = 0;
#endif // APP_FRAGMENTATION
    apsDataReq.radius = 0;                              // Usa o raio máximo possível.
    apsDataReq.APS_DataConf = APS_DataConf;               // Confirmação.

         APS_DataReq(&apsDataReq);
               //Chamada para o envio de informação.
}
```

```
//Callback do envio de dados.
static void APS_DataConf(APS_DataConf_t* confInfo)
{
	if(ZDO_SUCCESS_STATUS != confInfo->status)
	{
		contdata++;
		if(contdata==5)
		{
			BSP_OnLed(LED_YELLOW);
			HAL_StopAppTimer(&delayTimer);
			contdata=0;
			leaveNetwork();
		}

		else
		networkSendData();
	}

	else
	contdata=0;
}

//Função que trata o recebimento de dados. Stub
void APS_DataIndication(void)
{
}
/**************************************************************************
*************************************************
		Configurar parâmetros de sleep mode
**************************************************************************
*************************************************/

static void StartSleep(void)
{
	 zdoSleepReq.ZDO_SleepConf = ZDO_SleepConf;	// Confrimação de Sleep
	 ZDO_SleepReq(&zdoSleepReq);			//Inicialização do
modo de Sleep
}

static void ZDO_SleepConf(ZDO_SleepConf_t *conf)
{
 if (ZDO_SUCCESS_STATUS != conf->status)	// A pilha não conseguiu dormir
 SYS_PostTask(APL_TASK_ID);			// Chama a tarefa de
novo, para tentar dormir outra vez.
}

void ZDO_WakeUpInd(void)
{
	StartTimer();					//Inicializa a
rotina de temporização
```

```
        adcGetValue();
        //Inicializa a rotina de medição do ADC
}

/*****************************************************************************
************************************************
                Configurar parâmetros de medição de ADC
*****************************************************************************
************************************************/
 static void initAdcParams(void)
 {
        adcParams.callback = adcMeasured;
        //Função de confirmação de medição
        adcParams.resolution = RESOLUTION_8_BIT;                    //Resolução
da medição do ADC
        adcParams.sampleRate = ADC_4800SPS;                         //Taxa de
medição
        adcParams.bufferPointer = &(adcBuffer);                     //Ponteiro para o
buffer que detem os dados do ADC
        adcParams.selectionsAmount = 1;                             //Número de
medições por vez. Verifique o tamanho do Buffer para que seja compatível com o tamanho do
mesmo.
        adcParams.voltageReference = INTERNAL_1d1V;                 // Referência
para a medição do ADC.
 }

 static void adcGetValue(void)
 {
        //cont++;
        //if(cont == 100)
        //{
    //cont = 0;
        HAL_OpenAdc(&adcParams);
        //Inicializa o ADC
        HAL_ReadAdc(HAL_ADC_CHANNEL1);
                //Faz a leitura do canal escolhido
        //}
        //else
        //SYS_PostTask(APL_TASK_ID);
 }

 static void adcMeasured(void)
 {
        for(i=0; i<(sizeof(msgBuffer)-1); i++)                      //Prepara o
Buffer para o envio Wireless.
        appMessageBuffer.data[i] = msgBuffer[i];
        appMessageBuffer.data[i] = adcBuffer;
        appMessageBuffer.data[i+1] = ' ';
        apsDataReq.asduLength = sizeof(msgBuffer)+1;
        networkSendData();
```

```
        HAL_CloseAdc();
        //Desabilita o ADC, ajuda no baixo consumo de energia.
}

/************************************************************************************
*************************************************
                Task Handler Function
************************************************************************************
*************************************************/
//Tarefa da Aplicação. Leia na Bitcloud o funcionamento do escalonador de tarefas.
void APL_TaskHandler(void)
{
 switch (appState)
 {
  case APP_INITIAL_STATE:                       // Estado Inicial
        initAdcParams();
        initNetwork();
    SYS_PostTask(APL_TASK_ID);              // Próximo Passo
        break;

  case APP_NETWORK_JOINING_STATE:                       //Próximo Passo
        startNetwork();
        break;

  case APP_NETWORK_LEAVING_STATE:
                break;

        case APP_NETWORK_JOINED_STATE:                  //Inicializa o ciclo
                StartSleep();
                //StartTimer();
                break;
        default:
                break;
 }
}
/************************************************************************************
*************************************************
                Main Function
************************************************************************************
*************************************************/
//A Função main inicializa o micro controlador e a tarefa da aplicação.
int main(void)
{
 SYS_SysInit();

        for(;;)
 {
  SYS_RunTask();
 }
}
```

ANEXO III

Código Comentado do Tratador de Resultados

```
#include <stdio.h>

int main(void)
{
   FILE *fp;
   FILE *fp2;
   char c;
   long long int i = 0;
   long long int cont = 0;
   int cont2 = 0;


   fp = fopen("c:/users/lucas/desktop/Capture.txt", "r");      // Texto a ser lido com as
informações
   fp2 = fopen("c:/users/lucas/desktop/Bateria1.txt", "w"); // Arquivo texto de destino dos
dados tratados.

   while( i!=700000)                                          //Tamanho do Arquivo texto de
leitura.
  {
         c = fgetc(fp);
         if(cont2==1)
         printf("%d ",c&0x000000ff);
         if(c=='1')                                    //No do Sensor a Ser lido. e ter seus dados
         {                                                    // enviados para o arquivo destino
                cont++;
                cont2=1;
         }
      else
      cont2=0;

      i++;

   }
   printf("\n%d\n", cont);                                    //Imprime o número de dados.
   fclose(fp);

   return 0;
}
```

ANEXO IV

Modificações no programa para se trabalhar com OTAU

```
//Variáveis a Serem declaradas
static ZCL_Cluster_t otauCluster;
static ClusterId_t otauClusterId = OTAU_CLUSTER_ID;
static ZCL_OtauInitParams_t otauInitParams;
static ZCL_DeviceEndpoint_t otauClusterEndpoint;

//Definição de Função do Nó
#define OTAU_CLIENT
//ou
#define OTAU_SERVER

//Número de Clusters que serão acessados
#if defined(OTAU_CLIENT)
#define OUT_CLUSTERS_COUNT 1
#define IN_CLUSTERS_COUNT 0
#elif defined(OTAU_SERVER)
#define OUT_CLUSTERS_COUNT 0
#define IN_CLUSTERS_COUNT 1
#endif

//Funções que preparam os clusters para a gravação.
#if defined(OTAU_CLIENT)
otauCluster = ZCL_GetOtauClientCluster();
#elif defined(OTAU_SERVER)
otauCluster = ZCL_GetOtauServerCluster();
#endif

//Configuração de nó para o registro ZCL

otauClusterEndpoint.simpleDescriptor.endpoint = APP_OTAU_CLUSTER_ENDPOINT;
otauClusterEndpoint.simpleDescriptor.AppProfileId = PROFILE_ID_SMART_ENERGY;
otauClusterEndpoint.simpleDescriptor.AppDeviceId = WSNDEMO_DEVICE_ID;
```

```
otauClusterEndpoint.simpleDescriptor.AppInClustersCount = IN_CLUSTERS_COUNT;
otauClusterEndpoint.simpleDescriptor.AppOutClustersCount = OUT_CLUSTERS_COUNT;
#if defined(OTAU_CLIENT)
otauClusterEndpoint.simpleDescriptor.AppInClustersList = NULL;
otauClusterEndpoint.simpleDescriptor.AppOutClustersList = &otauClusterId;
otauClusterEndpoint.serverCluster = NULL;
otauClusterEndpoint.clientCluster = &otauCluster;
#elif defined(OTAU_SERVER)
otauClusterEndpoint.simpleDescriptor.AppInClustersList = &otauClusterId;
otauClusterEndpoint.simpleDescriptor.AppOutClustersList = NULL;
otauClusterEndpoint.serverCluster = &otauCluster;
otauClusterEndpoint.clientCluster = NULL;
#endif
ZCL_RegisterEndpoint(&otauClusterEndpoint);

//Definição do tipo de cluster
#if defined(OTAU_CLIENT)
otauInitParams.clusterSide = ZCL_CLIENT_CLUSTER_TYPE;
#elif defined(OTAU_SERVER)
otauInitParams.clusterSide = ZCL_SERVER_CLUSTER_TYPE;
#endif
otauInitParams.firmwareVersion.memAlloc =
APP_OTAU_SOFTWARE_VERSION;
otauInitParams.otauEndpoint = APP_OTAU_CLUSTER_ENDPOINT;
otauInitParams.profileId = PROFILE_ID_SMART_ENERGY;


//Função que ao finalizar o download de imagem, coordena o auto-reset.
static void otauClusterIndication(ZCL_OtauAction_t action)
{
if (OTAU_DEVICE_SHALL_CHANGE_IMAGE == action)
{
// Device has finished uploading image and can be reset. The
// application can perform additional actions here before the
// reset.
```

```
HAL_WarmReset();
}
}

//Função de Callback da Pilha
ZCL_StartOtauService(&otauInitParams, otauClusterIndication);
```

ANEXO V

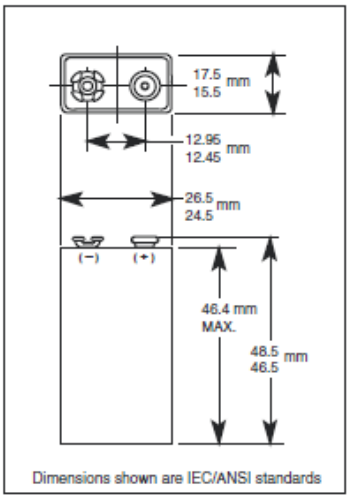
DataSheet da Bateria Utilizada nos Experimentos.

DURACELL®
COPPERTOP®

MN1604
Size: 9V (6LR61)
Alkaline-Manganese Dioxide Battery

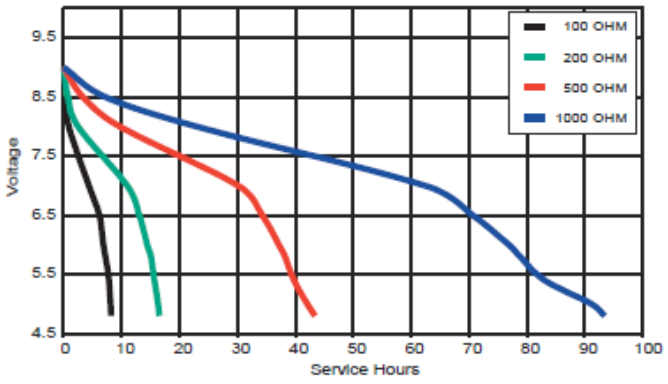
Zn/MnO_2

Nominal Voltage:	9 V
Operating Voltage:	9.6 - 4.8 V
Impedance:	1,700 m-ohm @ 1kHz
Typical Weight:	45 gm (1.6 oz.)
Typical Volume:	22.8 cm^3 (1.4 $in.^3$)
Terminals:	Miniature Snap
Storage Temperature Range:	-20°C to 35°C
Operating Temperature Range:	-20°C to 54°C (-4°F to 130°F)
ANSI: IEC:	1604A 6LR61



Dimensions shown are IEC/ANSI standards

TYPICAL DISCHARGE CHARACTERISTICS AT 21°C (70°F)



DURACELL® BATTERIES
Berkshire Corporate Park
Bethel, CT 06801 U.S.A.
Telephone: Toll-free 1-800-544-5454
Internet: www.duracell.com

* Delivered capacity is dependent on the applied load, operating temperature and cut-off voltage. Please refer to the charts and discharge data shown for examples of the energy / service life that the battery will provide for various load conditions.

This data is subject to change. Performance information is typical. Contact Duracell for the latest information

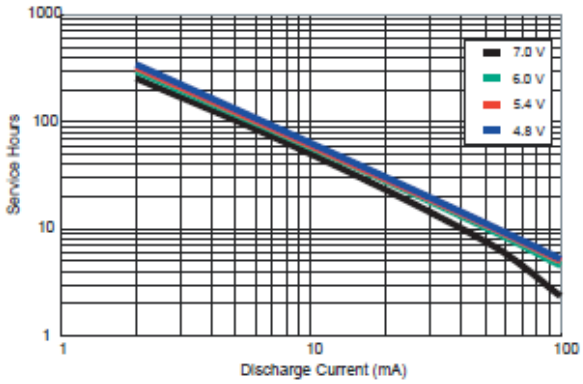
6/08

DURACELL®
COPPERTOP®

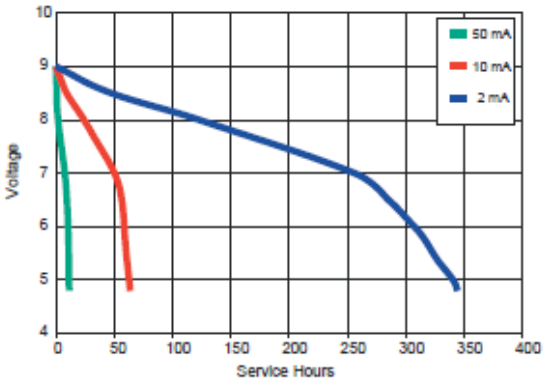
MN1604
Size: 9V (6LR61)
Alkaline-Manganese Dioxide Battery

Zn/MnO_2

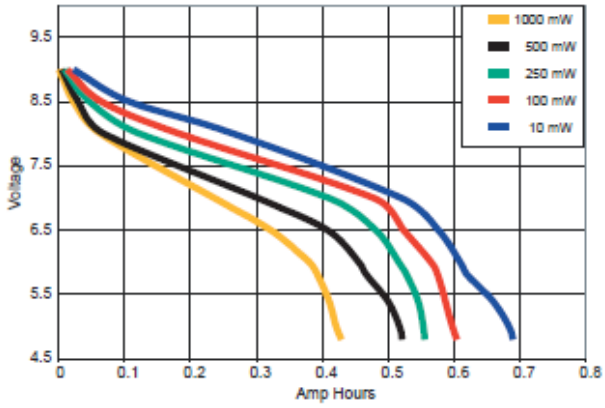
TYPICAL CONSTANT CURRENT DISCHARGE CHARACTERISTICS AT 21°C (70°F)



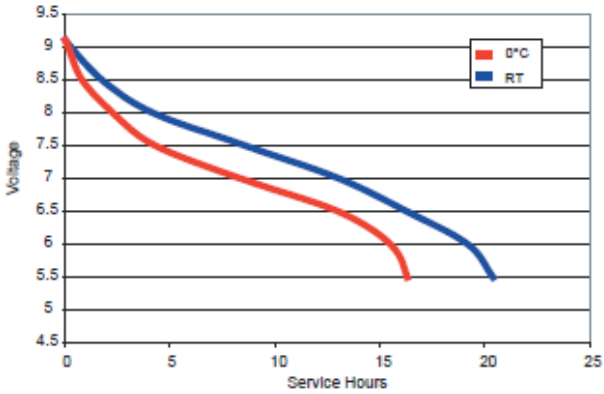
TYPICAL DISCHARGE CHARACTERISTICS AT 21°C (70°F)



TYPICAL DISCHARGE CHARACTERISTICS AT 21°C (70°F)



TYPICAL TOY TEST AT 21°C AND 0°C

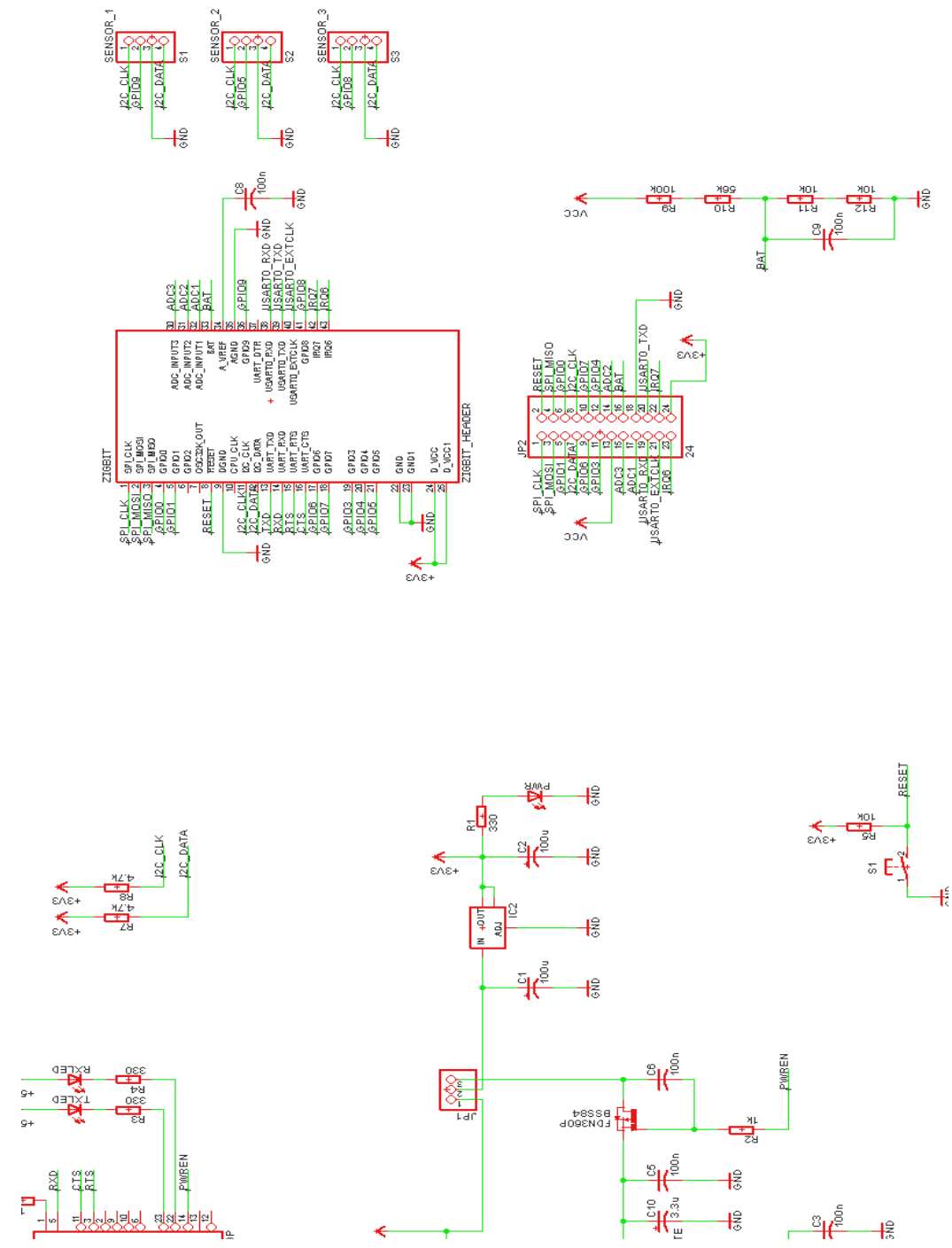


DURACELL®
BATTERIES
Berkshire Corporate Park
Bethel, CT 06801 U.S.A.
Telephone: Toll-free 1-800-544-5454
Internet: www.duracell.com

* **Delivered capacity is dependent on the applied load, operating temperature and cut-off voltage. Please refer to the charts and discharge data shown for examples of the energy / service life that the battery will provide for various load conditions.**

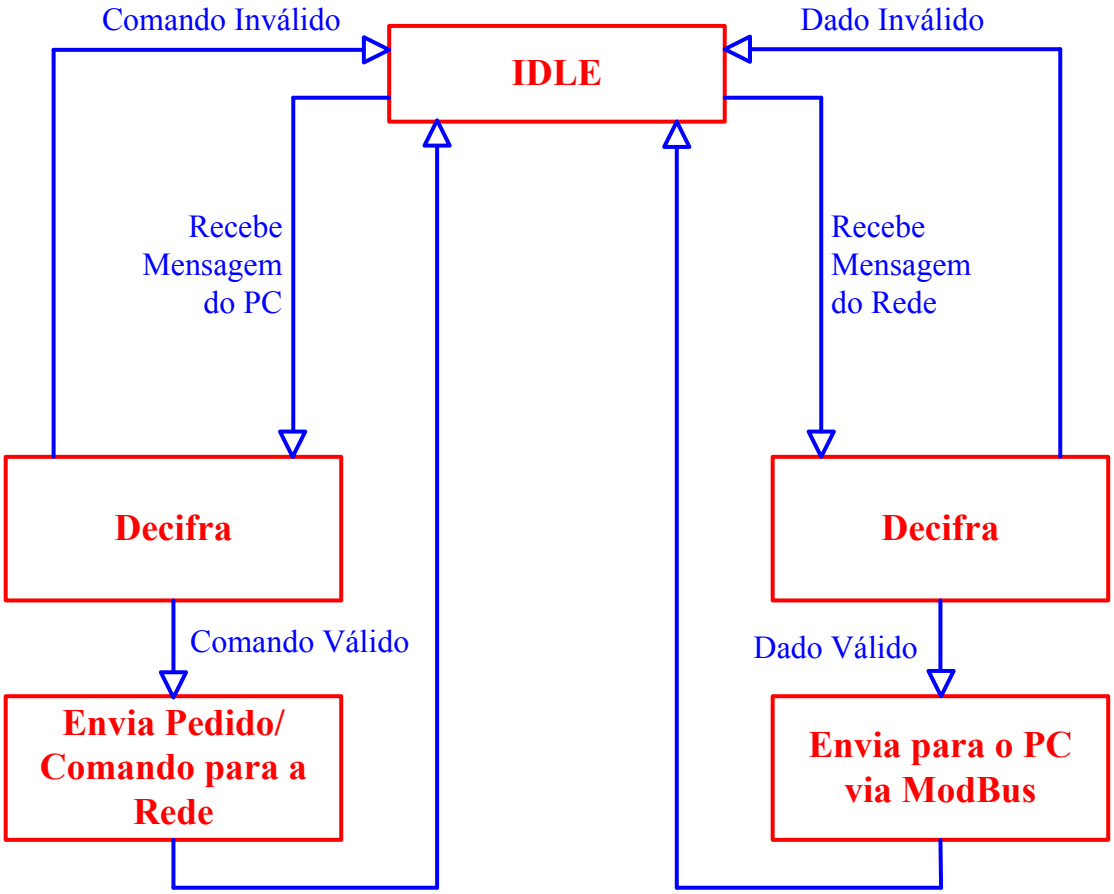
ANEXO VI

Esquemático da Placa ZigBit Channel 1.0



ANEXO VII

Item 1: Fluxograma do funcionamento sugerido para o Coordenador



Item 2: Fluxograma do funcionamento sugerido para o Atuador