UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

XARMBEE: UMA ARQUITETURA DE HARDWARE MODULAR MULTI-RÁDIO PARA GATEWAYS DE REDES DE SENSORES SEM FIO

ALEXANDRE LOPES MARTINIANO

MANAUS
2010

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALEXANDRE LOPES MARTINIANO

XARMBEE: UMA ARQUITETURA DE HARDWARE MODULAR MULTI-RÁDIO PARA GATEWAYS DE REDES DE SENSORES SEM FIO

Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Orientador: Prof. Dr. Carlos Maurício Seródio Figueiredo
Co-orientador: Prof. Dr. João Edgar Chaves Filho

MANAUS
2010

Ficha catalográfica elaborada pela Biblioteca Central da UFAM

M386x Martiniano, Alexandre Lopes
XARMBEE : uma arquitetura de hardware modular multi-rádio para gateways de redes de sensores sem fio / Alexandre Lopes Martiniano.- Manaus, AM : UFAM, 2010.
90 f. : il. color. ; 30 cm

Inclui referências.
Dissertação (Mestre em Engenharia Elétrica. Área de concentração: Controle e automação de sistemas). Universidade Federal do Amazonas. Orientador: Prof. Dr. Carlos Maurício Seródio Figueiredo.

1. Arquitetura de computador 2. Tecnologia Bluetooth 3. Rede de computador – Protocolos 4. Sistema global para comunicações móveis 5. General Packet Radio Service 7. Sistemas programáveis em chip 8. Sistemas de telecomunicação I. Figueiredo, Carlos Maurício Seródio (Orient.) II. Título

CDU (2007): 004.2(043.3)

ALEXANDRE LOPES MARTINIANO

XARMBEE: UMA ARQUITETURA DE HARDWARE MODULAR MULTI-RÁDIO PARA GATEWAYS DE REDES DE SENSORES SEM FIO

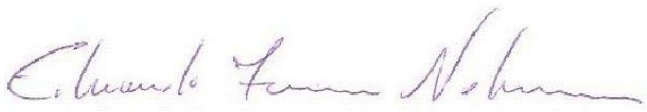
Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 1° de Julho de 2010.

BANCA EXAMINADORA

Prof. Dr. Carlos Maurício Seródio Figueiredo
Fundação Centro de Análise, Pesquisa e Inovação Tecnológica - FUCAPI

Prof. Dr. Horácio Antonio Braga Fernandes de Oliveira
Universidade Federal do Amazonas - UFAM



Prof. Dr. Eduardo Freire Nakamura
Fundação Centro de Análise, Pesquisa e Inovação Tecnológica - FUCAPI

Dedico esse trabalho aos meus pais Evandro e Izilda, pelo amor, carinho, compreensão e principalmente pelos valores ensinados que contribuíram para minha formação.

AGRADECIMENTOS

À Deus, por todas as conquistas alcançadas até hoje na minha vida e pela minha saúde.

À minha esposa Thamar e a minha princesa Ana Letícia, pela presença constante e incondicional, dando apoio nos momentos mais difíceis do desenvolvimento e elaboração deste trabalho.

Aos meus orientadores Prof. Dr. Carlos Maurício Seródio Figueiredo e Prof. Dr. João Edgar Chaves Filho pelo incentivo, paciência e muitas vezes pela compreensão com relação as minhas limitações. Meus sinceros agradecimentos.

Ao meu irmão Evandro Júnior, pelo incentivo e ajuda com críticas construtivas.

Aos meus colegas de mestrado, Alexandre Duarte, Antônio Santos, Charles Melo, Francisco Coelho, Gisele Lira, Luciano Lima, Luciano Pinto, Nairon Viana, Orlewilson Maia, Ricardo Erickson, Vandermi Silva, pela ajuda durante as disciplinas, momentos de descontração, sugestões, revisões de artigos, grupos de estudos, etc.

Aos professores do Programa de Pós-Graduação em Engenharia Elétrica da UFAM, pela dedicação e seriedade que desenvolvem seus trabalhos e orientações.

À Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM pelo apoio parcial fornecido este trabalho sob o projeto PIPT 1509/08.

Antes de entrar numa batalha, é preciso acreditar no motivo da luta.

(Sun Tzu - A Arte da Guerra)

RESUMO

Este trabalho apresenta uma arquitetura modular de *hardware* para viabilização e integração de diversas tecnologias de comunicação sem fio (*Bluetooth*, *GSM/GPRS* e *ZigBee*) em um *gateway* para Redes de Sensores Sem Fio (RSSFs). Desta forma, a proposta visa solucionar um problema pouco explorado de forma prática, que consiste na coleta presencial e remota de dados de redes geograficamente distribuídas. O objetivo é concentrar essa investigação em aspectos tecnológicos práticos que permitam a integração de diferentes tecnologias em uma solução prática e real. A arquitetura modular proposta contempla o gerenciamento de energia e a seleção da tecnologia de comunicação sem fio que será utilizada de acordo com a aplicação. Para validação da proposta um estudo de caso é realizado com o intuito de apresentar os resultados alcançados durante os experimentos realizados.

Palavras-Chave: Arquitetura, *Bluetooth*, *GSM/GPRS*, *Hardware*, RSSFs, *ZigBee*.

ABSTRACT

This work presents a modular architecture for hardware development and integration of various technologies of wireless communication (*Bluetooth*, *GSM / GPRS and ZigBee*) in a gateway for Wireless Sensor Networks (*WSNs*). Thus, the proposal seeks to address an issue not widely explored in a practical way that is the local and remote collection of data from geographically distributed networks. The objective of this research is to focus on practical technological aspects that allow the integration of different technologies in a practical and real hardware platform. The proposed architecture includes power management and selection of several technologies for wireless communication that will be used according to the application. To validate the proposed architecture case study is carried out in order to present the results achieved during the experiments.

Keywords: Architecture, Bluetooth, GSM/GPRS, Hardware, WSNs, ZigBee.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

ADC	*Analog to Digital Converter*
API	Application Programming Interface
CAD	*Computer-Aided Design*
CPLD	*Complex Programmable Logic Device*
CPU	*Central Processing Unit*
DAC	*Digital-to-Analog Converter*
DC	*Direct Current*
FPGA	*Field Programmable Gate Array*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
GSM/GPRS	*Global System Mobile/General Packet Radio Service*
HSDPA	*High-Speed Downlink Packet Access*
IAP	*In-Application Programming*
I^2C	*Inter-Intergrated Circuit*
IrDA	*Infrared Data Association*
ISP	*In-System Programming*
LDO	*Low Dropout Regulator*
PC	*Personal Computer*
PCMCIA	*Personal Computer Memory Card International Association*
PDA	*Personal Digital Assistants*
PLL	*Phase-Locked Loop*
PWM	*Pulse-Width Modulation*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
ROM	*Read-Only Memory*
RSSF	*Rede de Sensores Sem Fio*
RTC	*Real-Time Clock*
SD Card	*Secure Digital Card*
SDRAM	*Single Data Rate Sincronous Dynamic Random Access Memory*
SoC	*System-on Chip*
SPI	*Serial Peripheral Interface*
UART	*Universal Synchronous/Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VDC	*Voltage Direct Current*
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1 Introdução

Uma Rede de Sensores Sem Fio (RSSF) (AKYILDIZ *et al,* 2002) é uma rede formada por um grande número de sensores pequenos dispostos em uma determinada região para detectar e transmitir alguma característica física do ambiente (temperatura, pressão, etc). Em geral, os sensores encontram-se acoplados com elementos computacionais (microprocessadores) de pequenas dimensões e dispositivos de comunicação sem fio (rádios-transceptores) formando os nós sensores. O potencial de aplicação de tal arquitetura de monitoramento tem sido considerado em diversos contextos, tais como de monitoramento ambiental, militar e industrial (ARAMPATZIS *et al,* 2005), tornando-a uma área de pesquisa bastante ativa.

Um importante elemento dessas redes é o dispositivo que faz a integração das RSSFs com outras redes de comunicação. Estes elementos são denominados de *gateways*. Normalmente, esses dispositivos compartilham da rede sem fio para a coleta dos dados sensoriados e estão interligados a uma infra-estrutura computacional mais robusta de forma a permitir o devido armazenamento, processamento e consulta sobre os dados coletados. A *interface* entre o *gateway* e essa rede externa pode ser feita através de uma conexão *USB* com um computador ou por outras *interfaces* de redes a cabo ou sem fio que o dispositivo possa disponibilizar.

1.1 Motivação

Um problema pouco explorado de forma prática em RSSFs consiste na coleta remota de dados de redes geograficamente distribuídas. A questão é que para cobrir vastas áreas de monitoramento com uma única rede é inviável por requerer uma grande quantidade de dispositivos que promovam a interconexão sem fio ao longo de toda a área de interesse.

Desta forma, espera-se que RSSFs independentes sejam distribuídas geograficamente em sub-áreas de interesse, mas que de alguma forma dados coletados em cada uma possam ser concentrados para armazenamento e análise posterior. Para isso, existem diferentes estratégias para a conexão de um *gateway* de uma rede *ZigBee* com uma rede utilizando outra tecnologia de comunicação. Um cenário para essa aplicação seria vários pontos de coleta (nós sensores) situados em uma região metropolitana, enviando os dados monitorados para o *gateway* através de uma rede *ZigBee,* este por sua vez com acesso a uma rede celular *GSM/GPRS*, utilizando-se dessa tecnologia para o envio dos dados coletados a uma central. Em outros pontos de coleta sem a cobertura de uma rede celular, um operador desloca-se para uma determinada região próxima ao *gateway* para descarregar os dados em um computador ou dispositivo móvel para posterior transporte à central utilizando-se da tecnologia *Bluetooth.*

Um *gateway* de uma RSSFs deve prover os requisitos básicos como, por exemplo, agregação de dados dos sensores, duas vias para troca de dados, programação de nós sensores e gerenciamento de acesso para o usuário (PING *et al,* 2008). Diante dos trabalhos existentes na literatura, podem-se observar diversas arquiteturas para o projeto de *gateways,* alguns poucos se preocupam com a versatilidade na parte de comunicação do *gateway* e geralmente consideram uma única forma de comunicação deste com o mundo externo (SILVA *et al*, 2004). Outro problema é a forma de seleção eficiente entre múltiplos rádios-transceptores e os parâmetros utilizados na transmissão de forma a obter um menor consumo de energia.

A Figura 1.1 apresenta uma topologia típica de uma RSSF, onde informações são coletadas de uma rede utilizando o padrão de comunicação sem fio *ZigBee* e são encaminhadas (roteadas) até um ponto de concentração, um *gateway*, com ligação a redes externas (*GSM/GPRS e Bluetooth*) conectadas a *internet* que possuem maior capacidade para o armazenamento, processamento e disponibilização em um banco de dados (centralizador) para análise e disponibilidade global.

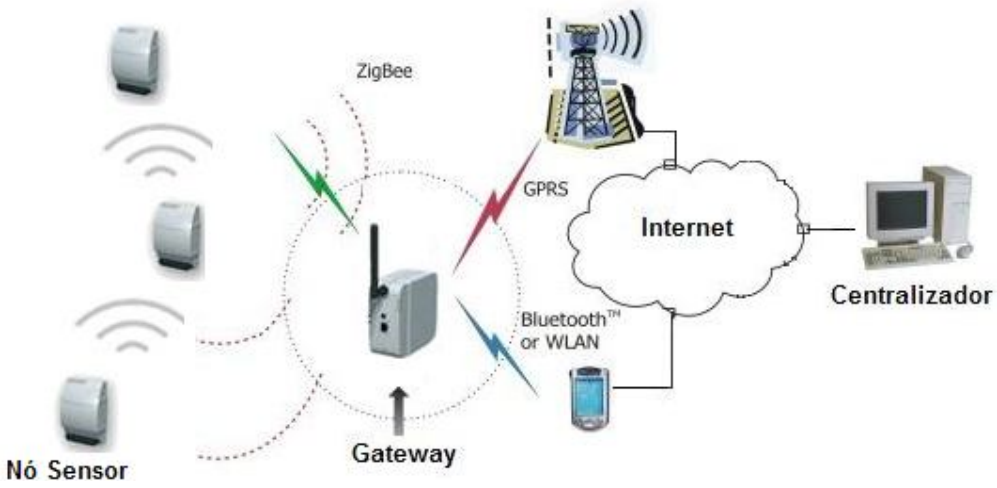


Figura 1.1 Topologia típica de uma Rede de Sensores Sem Fio.

Diante do problema considerado, a proposta deste trabalho consiste em investigar e propor uma arquitetura de *hardware* modular multi-rádio para *gateways* de RSSFs, ou seja, uma arquitetura com suporte às tecnologias de comunicação sem fio: *ZigBee*, *Bluetooth* e *GSM/GPRS*, permitindo a coleta de dados de diferentes RSSFs geograficamente distribuídas. O objetivo é concentrar essa investigação em aspectos tecnológicos práticos que permitam a integração dessas diferentes tecnologias em uma solução prática e real. A arquitetura proposta deve proporcionar a coleta de dados de RSSFs distribuídas, o gerenciamento de energia da plataforma de *hardware* e a seleção eficiente entre os rádios-transceptores de acordo com a aplicação e necessidade.

A coleta de dados será realizada de duas formas: (i) **Coleta presencial** e (ii) **Coleta remota**.

(i) **Coleta presencial**, onde um operador percorre cada rede remota com um dispositivo de coleta (*PDA* ou *notebook*) utilizando a tecnologia *Bluetooth* para a transferência dos dados coletados da rede para o dispositivo móvel e, posteriormente, realizar o descarregamento dos dados para o sistema centralizador.

(ii) **Coleta Remota**, onde um dispositivo de rede com tecnologia de comunicação de

longo alcance, como por exemplo, rede celular *GSM/GPRS*, é acoplado ao *gateway* de cada rede remota para o envio de dados diretamente ao sistema centralizador.

Embora mais prática, a coleta remota, está limitada ao monitoramento em áreas onde existe a disponibilidade da rede celular. Para flexibilizar essa ação, o *gateway* deve possibilitar diferentes *interfaces* de rádio.

O gerenciamento de energia será feito através do monitoramento do consumo de energia da plataforma de *hardware*, através de um resistor *shunt* adicionado em série com a fonte de alimentação que poderá ser uma fonte externa (adaptador *DC*) ou bateria. A partir dos dados de consumo de corrente o sistema selecionará o modo de operação de acordo com a aplicação e o rádio-transceptor mais adequado.

1.2 Objetivos

Os objetivos foram organizados da seguinte forma:

Objetivo Geral: Projetar, implementar e avaliar uma arquitetura modular multi-rádio para *gateways* de RSSFs.

Objetivos Específicos:

- Pesquisar na literatura soluções para a multiplexação e gerenciamento de uma estrutura multi-rádio em plataformas de *hardware* para RSSFs;
- Propor uma arquitetura de *hardware* que atenda às características e necessidades do projeto, levando em consideração os requisitos propostos como: a capacidade de processamento, sensoriamento, gerenciamento de energia e os rádios-transceptores necessários para comunicação com diversas tecnologias.
- Implementar a arquitetura de *hardware*, avaliar e realizar testes através de aplicações práticas e estudos de caso.
- Realizar avaliações do consumo de energia *versus* modo de funcionamento da arquitetura de *hardware*.

- Desenvolver uma *API* para configuração e troca de informações com o *gateway*.

1.3 Contribuições

As principais contribuições consistem em introduzir no projeto de *hardware* a capacidade de seleção entre a tecnologia de comunicação sem fio a ser utilizada dependendo da aplicação; a gerência de energia por módulo ativando quando necessário e desativando para economia de energia; a multiplexação de uma *interface* de comunicação para utilização de diferentes rádios-transceptores bem como a possibilidade de expansão e adaptação de outros novos.

1.4 Organização do Trabalho

O trabalho está organizado em cinco capítulos de forma a mostrar todas as etapas desenvolvidas para obtenção dos resultados apresentados, dos quais este é o primeiro. No Capítulo 2 apresenta-se uma revisão bibliográfica sobre as principais arquiteturas de nós sensores, arquitetura de *gateways* e algumas avaliações e métricas aplicadas em RSSFs que serão utilizadas como base para o projeto e avaliação da arquitetura proposta.

O Capítulo 3 descreve o princípio de funcionamento e o projeto da arquitetura, mostrando de forma sucinta a funcionalidade dos módulos e seus respectivos componentes. No final do capítulo apresenta-se também a arquitetura de *software* utilizada para o controle do *hardware*.

O Capítulo 4 descreve os testes realizados para mostrar o funcionamento da arquitetura utilizando cada rádio-transceptor e seus recursos em uma aplicação específica para cada caso. Descreve-se o método utilizado para a verificação do consumo de energia através da medição da corrente consumida pela arquitetura, quando está ativando e desativando cada rádio-transceptor e nos modos de funcionamento específicos.

São apresentados dois estudos de caso: o primeiro corresponde a Coleta Presencial de Dados e o segundo com a Coleta Remota de Dados, mostrando a capacidade da arquitetura proposta para solucionar os problemas apresentados.

Por fim, no Capítulo 5 são apresentadas as conclusões, sugestões para trabalhos futuros e publicações realizadas sobre o trabalho apresentado.

2 Fundamentos e Trabalhos Relacionados

2.1 Arquitetura de Nós Sensores

A construção de uma RSSF tem como principal requisito o projeto e a disposição dos nós sensores (KARL AND WILLIG, 2005), não deixando de suprir os requisitos específicos de uma determinada aplicação. Um nó sensor é basicamente constituído por uma unidade de energia, unidade de processamento, unidade de comunicação e uma unidade de sensoriamento.

Um dos primeiros projetos surgiu com uma família de nós sensores desenvolvidos por pesquisadores da Universidade da Califórnia, *Berkeley* (*UCB*) que são conhecidos como *Motes*. Esses nós sensores foram desenvolvidos tendo como principal objetivo o menor consumo de energia possível durante suas atividades (SILVA *et al*, 2004).

A primeira geração, implementada como projeto, tem como referência a tese de doutorado de *Seth Hollar* em 2000 (HOLLAR, 2000), é conhecida como *Macro Motes* ou *COTS Dust Motes*, em seguida uma segunda geração foi desenvolvida, os *Rene Motes* e finalmente, a última geração, formada pelos *MICA Motes* e *Smart Dust*.

Uma característica marcante dos *Motes* é a utilização de apenas um tipo de tecnologia de comunicação sem fio (*ZigBee)* e outra é o baixo poder de processamento, diminuindo o consumo de energia. Podemos citar algumas evoluções como a adaptação de microcontroladores com maior poder de processamento, embora mantendo o baixo consumo, e outras opções utilizadas na tecnologia de comunicação sem fio, como por exemplo, *Bluetooth*, *IrDA*.

Em alguns casos a utilização de memória *flash* de maior capacidade, e sistemas de gerenciamento de energia possibilitam que a arquitetura tenha maior autonomia.

Na Figura 2.1, verifica-se o crescimento em um curto espaço de tempo das tecnologias

aplicadas aos *Motes.* Pode-se conferir ainda a grande evolução na dimensão dos *Motes*, aumentando deste modo, a gama de aplicações da tecnologia.

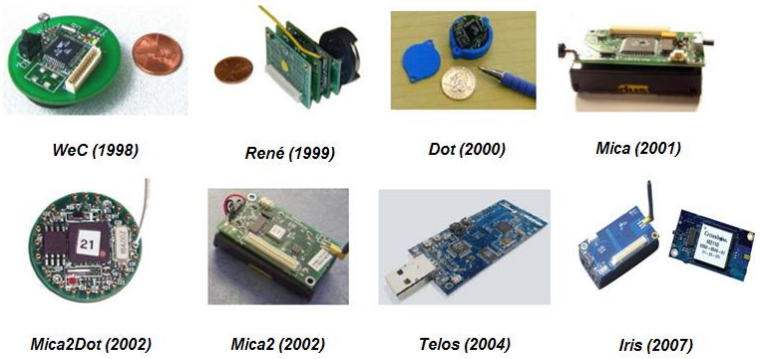


Figura 2.1 Evolução dos *Motes.*

O *ITRI ZBnode* (LEE AND HUANG, 2006), possui um microprocessador de *32 bits RISC*, memórias *SDRAM* e *Flash ROM* de *16 MB*, sendo um diferencial em relação a outras gerações de nós sensores. Possui também um sistema para gerenciamento de energia utilizando um *CPLD* que pode desabilitar a *CPU*, o rádio-transceptor ou o módulo de sensoriamento quando não estiverem ativos, otimizando o consumo de energia. A Figura 2.2 apresenta o diagrama em blocos do *ITRI ZBNode* evidenciando o sistema de gerenciamento de energia.

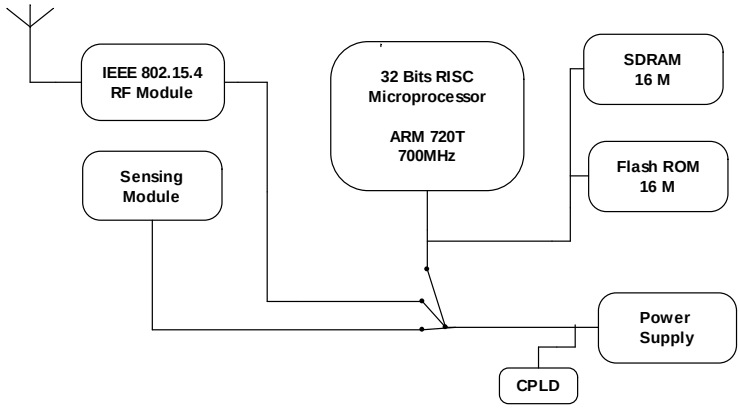


Figura 2.2 Diagrama em Blocos do *ITRI ZBnode.*

FONTE: Adaptado de (LEE AND HUANG, 2006).

Uma arquitetura apresentada em uma pesquisa recente (WEDDELL *et al*, 2009) mostra o projeto de um subsistema de energia reconfigurável para nós sensores autônomos. Esse novo sistema permite que módulos de energia sejam selecionados e colocados em ação de maneira automática sem a necessidade de uma readaptação ou modificação do *hardware*. O nó sensor pode monitorar e gerenciar de forma inteligente suas fontes de energia, analisando o nível de energia armazenada e a taxa de potência gerada. Formada pelo *SoC (System-on Chip) CC2430 OEM* (CC2430, 2009) da *Texas Instruments* e mais 4 módulos de energia: bateria, super-capacitor, placa fotovoltaica e adaptador *DC*, que são selecionados por uma chave analógica *AD708G 8-1*, contida no módulo multiplexador. A seleção é realizada pelo *SoC* provendo uma flexibilidade na escolha do tipo de alimentação para a arquitetura. A Figura 2.3 apresenta a arquitetura juntamente com os possíveis módulos de energia conectados ao módulo multiplexador. O diferencial do projeto é a seleção automática dos módulos, mas o que limita o mesmo é a capacidade de expansão que não apresenta muita flexibilidade.

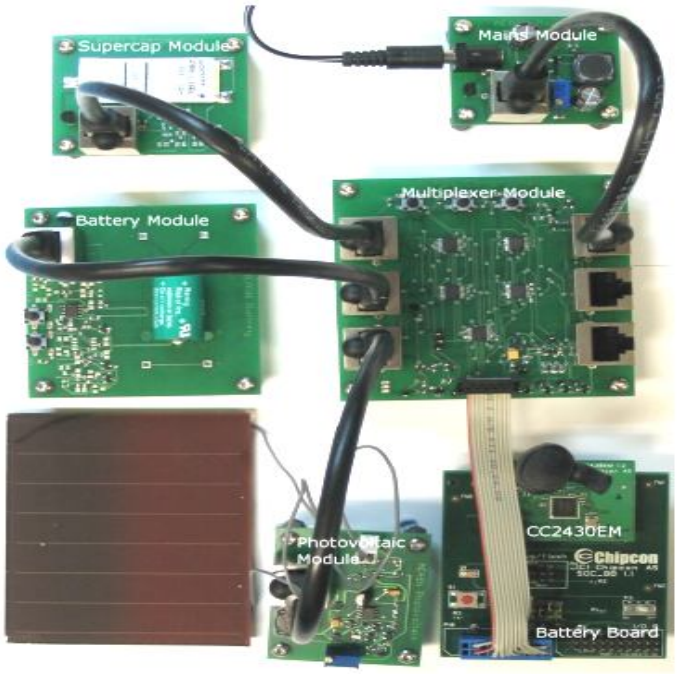


Figura 2.3 Arquitetura com Módulos de Energia.

FONTE: (WEDDELL *et al*, 2009).

O *BTnode* (BTNode, 2009), possui um microcontrolador *ATMega 128L; 8 MHz*; 51 pinos de (*I/O*); *RTC*; 8 canais *ADC* de 10 *Bits*; 6 canais *PWM* Programáveis (Resolução 2 a 16 *Bits*). O diferencial dessa arquitetura de *hardware* é a existência da possibilidade de chaveamento entre um sistema de transmissão e recepção de dados utilizando a tecnologia *ZigBee* ou *Bluetooth*, tornando o *BTNode* bastante versátil para aplicações em redes onde existe a necessidade de comunicação com elementos não pertencentes a rede, por exemplo, celulares, *PCs* entre outros. Outra vantagem do *BTnode* é o suporte ao *TinyOS*, mas possui seu próprio Sistema Operacional (S.O) o *BTNut.* O *BTNode* possui características de *gateway*, pois dependendo da aplicação poderá transmitir dados entre tecnologias de comunicação sem fio diferentes. A Figura 2.4 apresenta o diagrama em bloco do *BTNode* onde se pode verificar a seleção do rádio-transceptor de acordo com a aplicação.

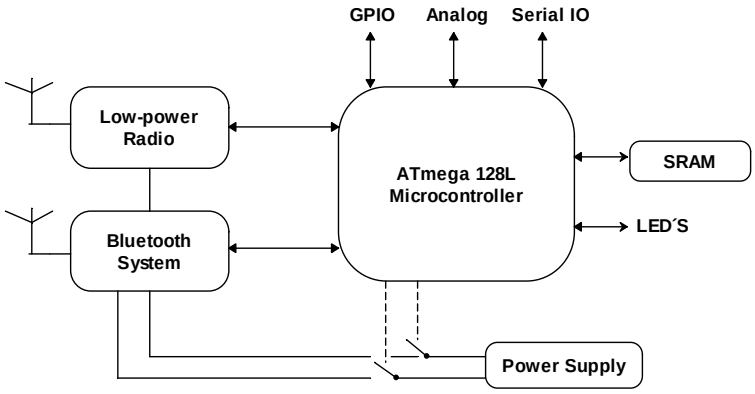


Figura 2.4 Diagrama em blocos do *BTNode.*

FONTE: Adaptado de: http://www.btnode.ethz.ch/. Acesso em Dez 09.

A *Multi-Radio Platform (*KOHVAKKA *et al*, 2006) é uma arquitetura que apresenta como diferencial a implementação em uma mesma estrutura de *hardware* de quatro rádio-transceptores, possibilitando desta forma uma comunicação mútua com outros quatro nós sensores, minimizando os atrasos (*hop delays*) existentes na rede. A arquitetura tem como núcleo principal de processamento um *FPGA Cyclone EP1C20* da *Altera* onde quatro *softcores NIOS II* são sintetizados, proporcionando uma reconfigurabilidade que permite uma

otimização da arquitetura. A plataforma pode atuar sozinha em uma RSSF, mas pode conectar-se a um computador através de uma *interface RS-232* (serial) para monitoramento das operações internas. Tem como fonte de energia quatro pilhas (*4 x AA*) ou um adaptador externo de tensão *DC*, que fornece os níveis de 1.5V e 3.3V respectivamente através de reguladores de tensão.

A arquitetura pode ser aplicada como roteador, *gateway* ou nó sensor de alto desempenho. O módulo de comunicação utilizado é o *nRF2401A* da *Nordic Semiconductor* que opera na frequência de *2.4GHz* e possui uma taxa de 1 *Mbps*. A Figura 2.5 apresenta o diagrama em blocos da *Multi-Radio Platform*, onde se pode verificar o núcleo principal de processamento (*FPGA*) e os quatro rádios-tranceptores em evidência.

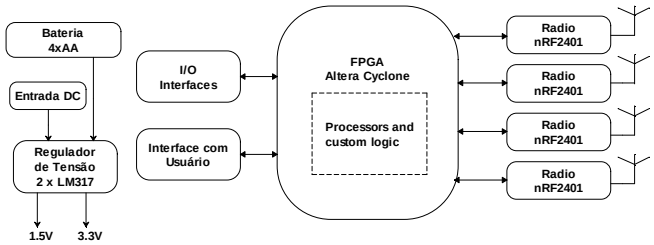


Figura 2.5 Diagrama em Blocos da *Multi-Radio Platform.*

FONTE: Adaptado de (KOHVAKKA *et al*, 2006).

A *Multi-Radio Platform* é composta por três tipos diferentes de placas chamadas de: *mother board*; *radio board* e *I/O board*. A plataforma completa é apresentada na Figura 2.6 abaixo com a especificação dos principais componentes.

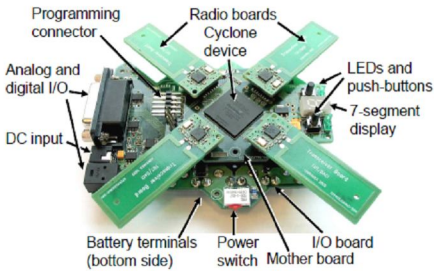


Figura 2.6 *Multi-Radio Platform.*

FONTE: Retirado de (KOHVAKKA *et al*, 2006).

Todas essas arquiteturas de nós sensores apresentam características de *gateway* e podem ser utilizadas com esse intuito, servindo como base para melhorias e desenvolvimentos futuros. Verifica-se em cada trabalho apresentado as diferentes características com relação a multiplexação e otimização dos recursos de *hardware*, proporcionando subsídios para o projeto da arquitetura proposta.

2.2 Arquitetura de *Gateways*

O *SPB400 – Stargate Gateway* (STARGATE, 2004), da *Crossbow* é um computador de pequenas dimensões baseado no S.O *Linux* que possui pontos de acesso para conexão de uma RSSF a outra de maior capacidade sem a necessidade de um computador ou servidor. Múltiplas *interfaces* como *Ethernet, USB* e Serial são disponíveis, até *slots PCMCIA* e *Compact Flash*, propiciando conexões adicionais, como *Wi-Fi*. O *SPB400* não possui uma flexibilidade para adaptação de novos módulos de comunicação sem fio, pois possui uma estrutura rígida. Outro ponto importante é a parte de gerenciamento de energia, que necessita de uma fonte externa para que o *gateway* possa exercer todas as suas funcionalidades.

O *NB100 – Stargate NetBridge* (NB100, 2007), da *Crossbow* é uma versão atualizada do *SPB400 – Stargate Gateway*, possui um processador *Intel IXP420 XScale*, é equipado com memória *flash* de *8 MB, 32 MB* de *RAM* e um disco de sistema de *2 GB USB 2.0*. Possui *interface* de comunicação *ethernet*. O *NB100* tem como diferencial o alto poder de processamento e armazenamento de dados, mas possui os pontos fracos citados para o *SPB400*.

O *Uber Board – Development Kit* (UBER BOARD, 2009) é uma plataforma de desenvolvimento de comunicação sem fio (*Bluetooth* e *GSM/GPRS*) comercial que possui características para testes de integração com redes *ZigBee*. Sendo uma plataforma de testes, não possui gerenciamento de energia. Tem como diferencial o método utilizado para multiplexar uma *interface* de comunicação, possibilitando uma flexibilidade limitada para

expansão de módulos de comunicação sem fio. Não possui gerenciamento de energia necessitando de um adaptador *DC* (fonte externa) para o seu funcionamento.

O *Waspmote* (WASPMOTE, 2010) é uma arquitetura baseada em uma estrutura modular. A idéia é integrar somente os módulos necessários para uma determinada aplicação. Os módulos suportados são: *ZigBee, GSM/GPRS, GPS*, *Sensor* e *SD Card*. Tem um microcontrolador *ATmega 1281* que é o núcleo da unidade de processamento e gerencia toda a arquitetura. Tem como diferencial a multiplexação da *UART* para comunicação com diversos módulos, mas o gerenciamento de energia para cada módulo e a expansão da *interface* de comunicação não são implementados na arquitetura.

Todas as arquiteturas apresentadas são comerciais, e sempre interligam uma RSSF a *Internet* através de uma tecnologia de comunicação rígida geralmente uma conexão *ethernet* ou sem controle sobre o gerenciamento de energia, fundamental em RSSFs. A Figura 2.7 apresenta as arquiteturas comerciais de *gateways*.

Figura 2.7 Arquiteturas Comerciais de *Gateways*.

Em (JIN *et al*, 2009) é apresentado o projeto de um *gateway* utilizado para interconexão entre uma RSSF com uma rede *Wi-Fi*. São apresentadas estratégias para redução do custo do ciclo de trabalho através do particionamento do sistema (modularização) e a redução do tempo necessário para ativar e desativar os componentes que consumem maior quantidade de potência. Foi realizado um estudo comparativo entre o *gateway* e o *NB100-Stargate* para comprovação da eficácia do sistema. Um protótipo foi construído, e

experimentos realizados comprovaram uma redução significativa na latência e no consumo médio de energia comparado ao *NB100-Stargate*.

2.3 Avaliações e Métricas para RSSFs

Devido às diversas arquiteturas existentes, há grandes dificuldades em comparar arquiteturas de *hardware* para RSSFs pelo fato da grande quantidade de aplicações e das características distintas de cada plataforma (BEUTEL, 2006). Com essa motivação o trabalho apresenta um conjunto de métricas padrões para as plataformas, baseadas em uma metodologia utilizando gráficos de estrelas multidimensionais para uma comparação entre plataformas, na fase de projeto, de fácil utilização e visualmente intuitiva. Dessa forma podem-se analisar todas as plataformas em um mesmo gráfico verificando as vantagens que cada uma possui em relação às outras, ou seja, dependendo da necessidade do projeto, qual seria a melhor escolha como plataforma de *hardware*. A Figura 2.8 apresenta um exemplo de um gráfico estrela.

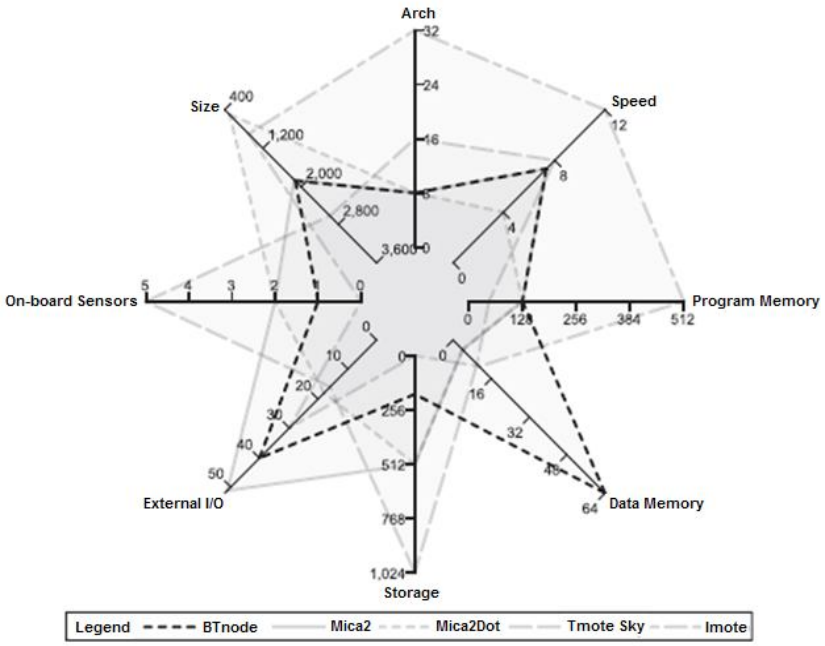


Figura 2.8 Gráfico Estrela Multidimensional.

Fonte: Retirado de (BEUTEL, 2006).

O gráfico da Figura 2.8 apresenta as seguintes plataformas: *BTNode, Mica2, Mica2Dot, Tmote Sky e Imote* que são analisadas em relação as seguintes características: tamanho, arquitetura, velocidade, memória de programa, memória de dados, armazenamento, pinos de entrada/saída (*I/O)* externos, sensores na placa. Pode-se dessa forma verificar vantagens de uma plataforma em relação à outra, ou seja, escolher qual seria a melhor para projetos específicos.

Modelos para a predição do consumo de corrente em arquiteturas de nós sensores parecem ser ainda uma questão em aberto. De fato os modelos para avaliação do consumo de corrente sempre expressam o consumo em função de equações com um grande número de parâmetros. Esses parâmetros precisam ser extraídos de dados que muitas vezes não estão disponíveis nas folhas de dados (*datasheets*) dos componentes eletrônicos. A caracterização do *hardware* é a única forma para contornar o problema. De acordo com essa premissa o trabalho analisado (BARBONI AND VALLE, 2008) apresenta um sistema baseado em uma placa eletrônica dedicada para verificação do consumo de corrente, composta pelos seguintes dispositivos: *High Side Current Shunt Monitor INA139* (INA138,1999) e o *Coulomb Counter LTC4150* (COULOMB, 2005), apresentando resultados satisfatórios na avaliação do consumo de energia de uma plataforma de *hardware*.

O *MATSNL* (MATSNL, 2008) é um *Toolbox* para o *software MATLAB* que realiza o cálculo do tempo de vida de um nó sensor em relação à potência da bateria; e consequentemente a escolha de determinadas arquiteturas de nós sensores para aplicações específicas. O *Toolbox* foi desenvolvido com o intuito de ser uma ferramenta de fácil utilização e modificação, possibilitando a combinação entre diferentes rádios-transceptores, processadores e sensores, expondo deficiências no desenvolvimento do projeto em um estágio inicial.

Pode-se verificar alguns fatores diferencias que não estão presentes nos diversos trabalhos citados como: a modularidade na adaptação de novos componentes de *hardware*,

multiplexação de uma *interface* para comunicação com diversos rádios-transceptores, gerenciamento de energia utilizando um método de determinação de tempos para ativação e desativação de componentes que consumem maior quantidade de energia, técnicas para o monitoramento do consumo de corrente da arquitetura e do cálculo para obtenção do valor da energia consumida.

Por fim, verifica-se a grande dificuldade existente na comparação entre diversas arquiteturas de *hardware*, e obtêm-se algumas ferramentas computacionais que auxiliam na predição e análise dos parâmetros obtidos através de experimentos práticos que ajudam na caracterização da arquitetura, mostrando suas vantagens e desvantagens em relação a outras existentes, de tal forma que se consiga reconhecer quais os pontos que devem ser aprimorados ou até mesmo desenvolvidos.

3 *XArmBee* - Arquitetura, Projeto e Implementação

Este trabalho propõe a arquitetura de um dispositivo de *hardware* adequado à função de *gateway*. O mesmo considera como principal requisito a possibilidade de adaptação modular de diferentes tipos de comunicação de forma a possibilitar diferentes estratégias de coleta de dados de uma rede.

Embora em RSSFs a economia de energia seja sempre um requisito fundamental, considera-se que o *gateway* deva atendê-lo da melhor forma possível mesmo com a necessidade de possuir uma arquitetura mais robusta. Dessa forma, o *gateway* deve aliar poder de processamento, armazenamento e comunicação com uma estratégia de gerência de energia.

3.1 Arquitetura de *Hardware*

O modelo funcional proposto para o *gateway* é composto basicamente por três unidades: (i) Unidade Energia, responsável pela alimentação de todos os módulos do sistema; (ii) Unidade Processamento e Armazenamento de Dados, módulo central de controle de todo o sistema; e (iii) Unidade de Comunicação, responsável pela *interface* com cada módulo de comunicação. A Figura 3.1 apresenta o diagrama em blocos do modelo funcional. Suas descrições detalhadas são apresentadas nas seções subsequentes.

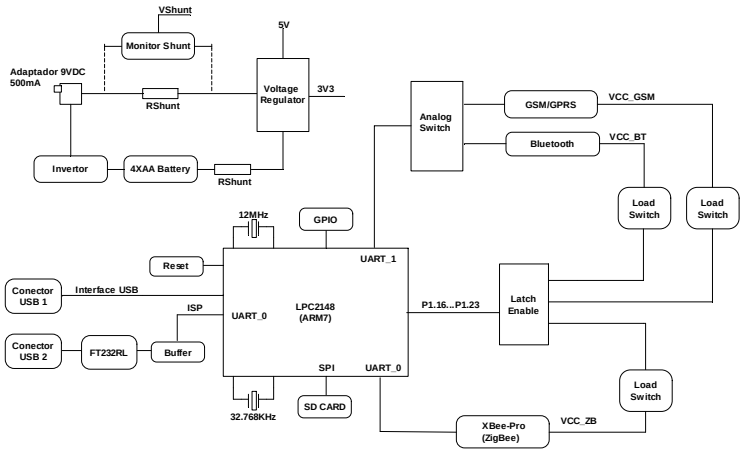


Figura 3.1 Diagrama em Blocos da Arquitetura *XArmBee*.

A Figura 3.2 apresenta a implementação do protótipo funcional da arquitetura *XArmBee*, destacando cada uma de suas unidades e o Gravador ISP, responsável pela atualização do *firmware* da arquitetura. A estrutura modular foi desenvolvida utilizando-se placas circulares montadas sobrepostas em camadas, sendo a primeira camada a Unidade Energia, a segunda Unidade Processamento e Armazenamento de Dados e a última camada a Comunicação.

A montagem em camadas facilita a substituição e, até mesmo, a adaptação de novos módulos ou expansão da capacidade da arquitetura.

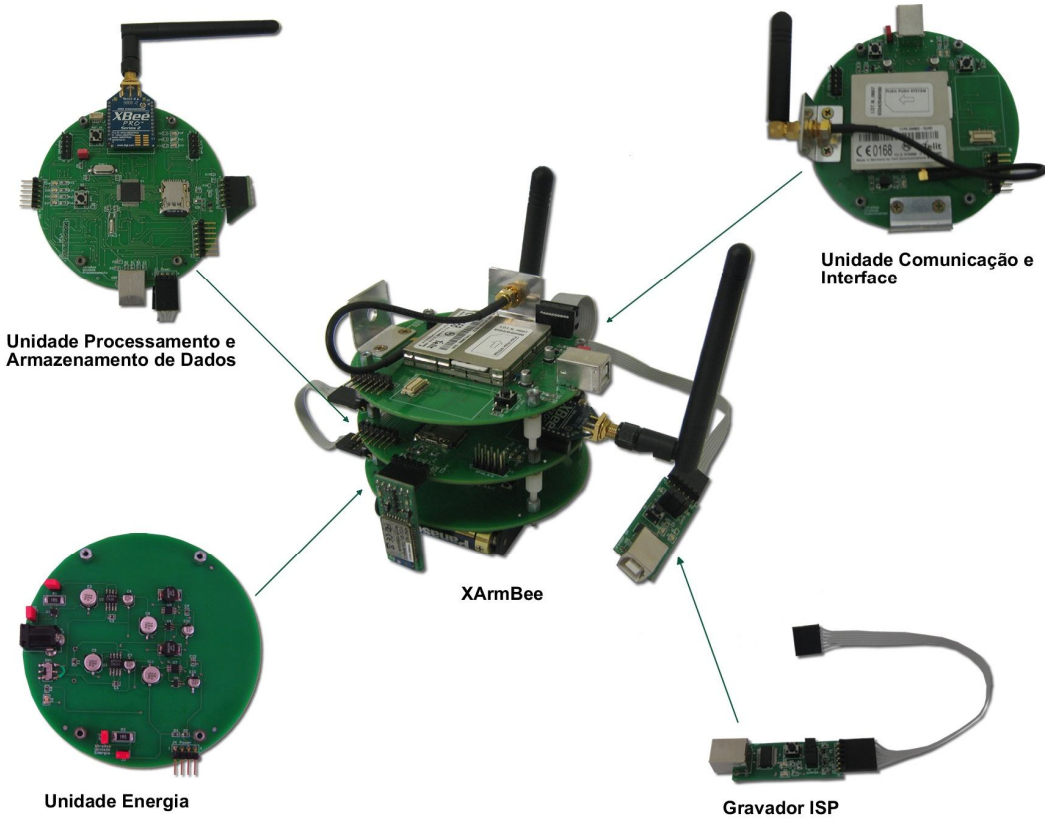


Figura 3.2 Componentes da Arquitetura *XArmBee*.

3.2 Procedimento Geral de Projeto e Implementação

No projeto da arquitetura, todas as placas de circuito impresso (PCIs) foram desenvolvidas utilizando-se o *software ORCAD PCB DESIGN SUITE* 16 (ORCAD, 2008), que é composto por um conjunto de ferramentas para o desenvolvimento de esquemático

(*Capture CIS*), *layout* de PCI (*Layout Plus*), roteamento automático, entre outras funções. O processo de desenvolvimento utilizado no projeto seguiu três etapas distintas: (i) Criação do esquemático; (ii) Criação do *Layout* das PCIs e (iii) Geração dos arquivos de fabricação (*Gerbers*).

O esquemático foi desenvolvido com a ferramenta *Capture CIS* (ORCAD, 2008) que possui um ambiente intuitivo e de fácil interação. Com esta ferramenta, realiza-se a interconexão entre os componentes que serão utilizados no projeto. Grande parte dos componentes existe no banco de bibliotecas (*library*) da própria ferramenta, mas no caso de projetos específicos que necessitam de componentes de fabricantes que não estão na biblioteca, a ferramenta permite que o projetista possa criar os componentes de acordo as especificações técnicas (*datasheets)* dos fabricantes. A Figura 3.3 apresenta o esquemático do Gravador ISP, descrito posteriormente, que é utilizado para atualização do *firmware* do microcontrolador.

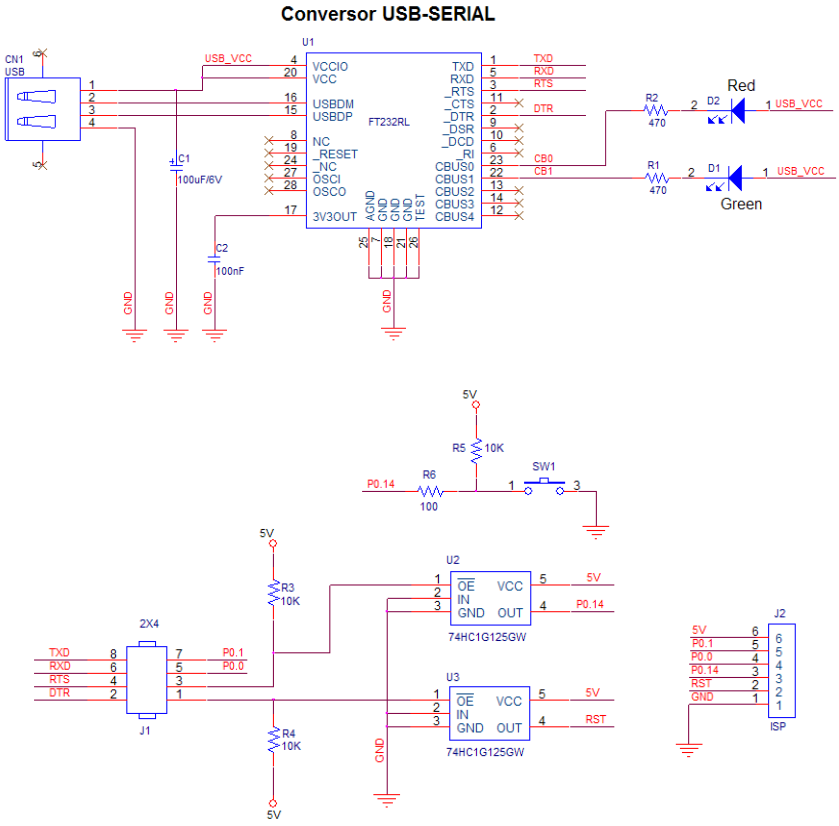


Figura 3.3 Esquemático do Gravador ISP.

Após a conclusão do esquemático cada componente deve ser associado ao seu *footprint* correspondente, ou seja, o desenho de suas dimensões reais que devem ser impressas em uma PCI, da mesma forma que ocorre com os componentes do esquemático, grande parte dos *footprints* encontra-se em bibliotecas da própria ferramenta, mas existe a possibilidade também da criação de novos *footprints* de acordo com a necessidade. A ferramenta utilizada para criação dos *footprints* é o *Layout Plus* (ORCAD, 2008).

Após ser associado, cada componente do esquemático com seu respectivo *footprint* utiliza-se o *Layout Plus* para realização do roteamento da placa, ou seja, colocar os *footprint´s* em uma planilha *CAD* com as respectivas ligações que deverão ser transformadas em trilhas, podendo ser dispostas em várias camadas de acordo com o projeto. Todas as placas desenvolvidas foram feitas em duas camadas: *TOP* (Superior) e *BOTTOM* (Inferior). A Figura 3.4 apresenta a PCI do Gravador ISP no *Layout Plus* já realizado o roteamento e o *layout* completo da placa.

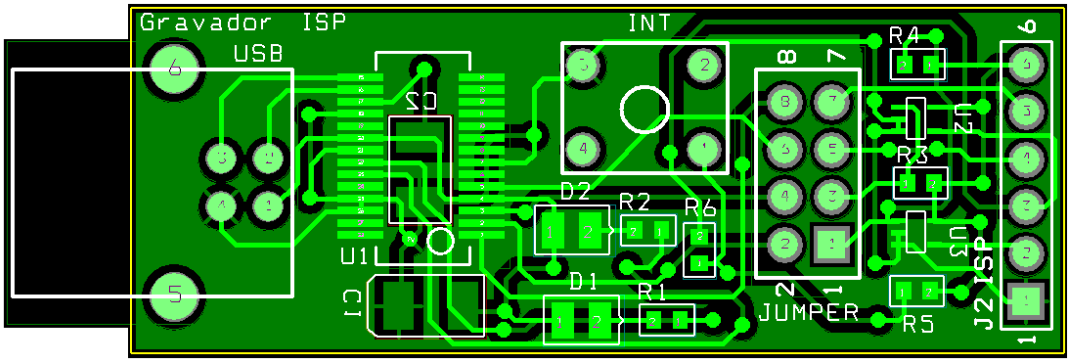


Figura 3.4 *Layout* da Placa do Gravador ISP.

A última etapa do processo de desenvolvimento é a geração dos arquivos de fabricação (*Gerbers*), realizada na própria ferramenta *Layout Plus.* O conjunto de arquivos gerados no formato padrão *Gerber RS-274D* são enviados a uma empresa de prototipação de PCIs que os recebe e fabrica na quantidade especificada pelo projetista seguindo fielmente o que foi projeto no *Layout Plus.*

3.3 Unidade Energia

O *gateway* através da Unidade Energia pode ter sua alimentação feita através de uma fonte externa de *9 VDC/500mA* ou através de uma bateria formada por quatros pilhas do tipo *AA*. O esquemático da unidade é apresentado no Apêndice A.2.

Analisando o diagrama em blocos apresentado na Figura 3.1, verifica-se que quando a alimentação é realizada por uma fonte externa o sinal de entrada de *9 VDC* é aplicado ao sistema. Esse sinal irá para a entrada do regulador de tensão de *5 VDC,* e sua saída para outro regulador de *3V3,* gerando desta forma os níveis de tensões *5 VDC* e *3.3 VDC* necessários para o funcionamento do sistema. Para o monitoramento do consumo de corrente, um resistor *shunt* foi adicionado juntamente com um *monitor shunt INA138* (INA138, 1999*)*. O *INA138* basicamente amplifica o sinal de tensão sobre o resistor *shunt* conectado em série com a fonte de alimentação. A tensão medida *VShunt* é proporcional ao nível de corrente consumida pelo sistema. A saída de tensão analógica pode ser facilmente visualizada em um osciloscópio com uma taxa de amostragem de *4 GSa/s* e *1 GHz* de banda ou medido através de um multímetro de precisão.

Quando a fonte externa é desconectada do sistema, o sinal de *9 VDC* tem seu nível de tensão igual a zero, habilitando através de dois inversores a alimentação do sistema por uma bateria formada por quatro pilhas *AA*. Uma vez habilitada, a bateria fornece uma tensão de *6 VDC* aplicada a dois conversores *DC-DC* do tipo *Boost,* que fornecem os níveis de tensões *5 VDC* e *3.3 VDC* mantendo o funcionamento do sistema.

A medição do consumo de corrente também pode ser realizada do mesmo modo, pois um resistor *shunt* foi adicionado em série com a bateria. A principal característica da unidade é o chaveamento automático do modo de alimentação, o gerenciamento dos níveis de tensões e a possibilidade de monitoramento do consumo de energia do sistema.

A Figura 3.5 apresenta o aspecto final da Placa Unidade Energia.

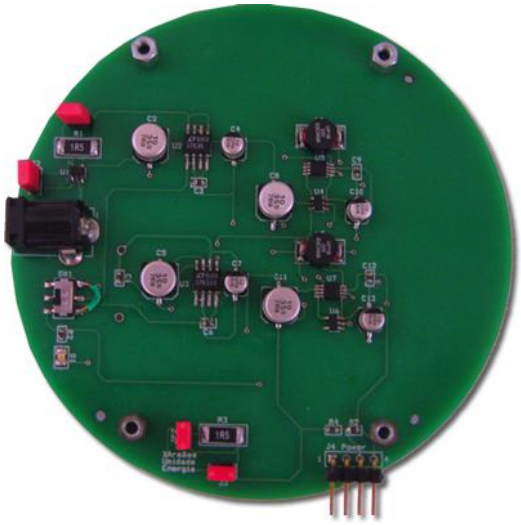


Figura 3.5 Placa Unidade Energia.

3.4 Unidade Processamento e Armazenamento de Dados

A Unidade Processamento e Armazenamento de Dados é o núcleo principal da arquitetura, é responsável pelo monitoramento, processamento e gerenciamento de todo o sistema. O esquemático da unidade é apresentado no Apêndice A.3.

A unidade é composta pelo microcontrolador *LPC2148 (ARM7TDMI-S)*(NXP, 2005) da *Philips* de *32 bits* com *512 KB* de memória *Flash*, *32 KB* de memória *RAM*, suporte a *interface ISP/IAP* via *software bootloader* contido no próprio circuito integrado. Possui controlador *USB 2.0 Full-Speed* integrado. Possui diversas *interfaces* de comunicação como: 2 *UARTs, SPI,* e *I2C,* 2 conversores *ADCs* com 14 canais de *10 Bits* cada, 1 conversor *DAC, PWM* e 45 linhas de *GPIO*. Frequência máxima de *60 MHz* de operação da *CPU*, programável através de *PLL* integrado e possui tensão de operação na faixa de *3.0 VDC* a *3.6 VDC*. Com todas essas características o microcontrolador é indicado para o desenvolvimento de *gateways* de comunicações e conversores de protocolos.

O princípio de funcionamento da unidade é baseado no recebimento de dados monitorados de uma rede *ZigBee* através do módulo *XBee-PRO* conectado à *UART_0* e armazenados em um *SD Card* de *2 GB* para de acordo com requisições, aleatórias ou em

períodos definidos do dia, esses dados possam ser transferidos para um banco de dados centralizador através de *interfaces* de comunicação (*Bluetooth* ou *GSM/GPRS*) que estão multiplexadas através da *UART_1* para posterior análise. Para o seu correto funcionamento, a unidade precisa de um *clock* de *12 MHz* para gerar a frequência de operação da *CPU* que pode ser um valor múltiplo de 12 definido pelo *PLL* interno e limitando-se ao valor máximo de *60 MHz*. Necessita também de um *clock* de *32.768 KHz* para o *RTC* interno utilizado para ativar o sistema em tempos definidos. A atualização do *firmware* do sistema é realizada através da *interface ISP (In-System Programming),* ou seja, a atualização do *firmware* é realizada no próprio sistema. Existe também um barramento de controle (*Latch tipo D - 74HC573*) para gerenciar a ativação dos módulos de comunicação. Tudo feito através de pinos dedicados do microcontrolador para essas funções. A unidade ainda possui uma *interface USB 2.0* e pinos de expansão (*GPIO*) para conexão de sensores/atuadores. Dois rádios-transceptores estão na unidade que são: *XBee-PRO (ZigBee)* e o *BlueSmirf (Bluetooth)*. A Figura 3.6 apresenta a Placa Unidade Processamento e Armazenamento de Dados.

Figura 3.6 Placa Unidade Processamento e Armazenamento de Dados.

3.4.1 Gravador ISP

A placa Gravador ISP (esquemático apresentado no Apêndice A.1) tem como componente principal um circuito integrado conversor *USB/Serial FT232RL*, responsável pelo fornecimento de todos sinais de uma porta serial convencional, embora somente os sinais *Rx* e *Tx* sejam utilizados para atualização do *firmware* do microcontrolador. Um circuito de controle de estados também é utilizado com a função de automatizar o processo de Programação em Sistema (*In-System Programming*). A Figura 3.7 apresenta a placa Gravador ISP.

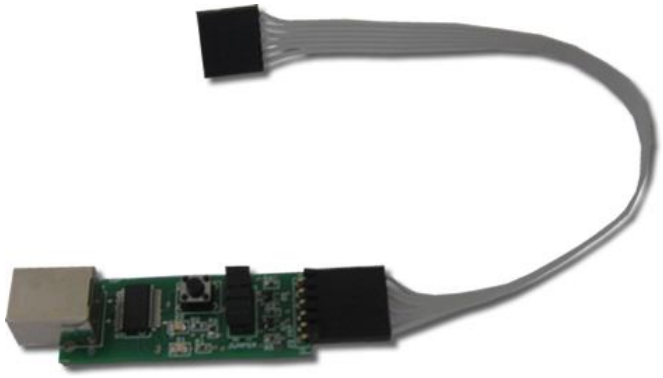


Figura 3.7 Placa Gravador ISP.

3.5 Unidade de Comunicação

A Unidade de Comunicação (esquemático apresentado no Apêndice A.4) é responsável pela seleção do rádio-transceptor que será utilizado na comunicação sem fio, atribuindo dessa forma características para o *gateway* de comunicação multi-rádio. A unidade integra o módulo: *Telit GM862-QUAD (GSM/GPRS)* que juntamente com os módulos *XBee – PRO (ZigBee)* e *BlueSmirf (Bluetooth)* foram selecionados para a implementação inicial do sistema, pois possuem como *interface* de comunicação o padrão *UART (RS-232)*, necessário para o *interfaceamento* com o microcontrolador *ARM7* e por possuírem um baixo consumo de energia, vale ressaltar que todos os aspectos de *interface* permitem a fácil

integração e simultânea de quaisquer módulos de comunicação alternativos, com gerência de energia individual, sendo essa a principal característica diferencial do projeto.

A unidade tem como princípio de funcionamento a ativação do rádio-transceptor através do barramento de controle, posteriormente a seleção da porta *UART* a ser utilizada realizada através do chaveador analógico *TS3A5017* e por fim a transferência de dados do *SD Card* da unidade de processamento e armazenamento de dados para o rádio-transceptor desejado. Com isso temos o gerenciamento da energia utilizada por cada rádio-transceptor definindo o tempo que o mesmo permanece ativado e desativado. O sinal de ativação e desativação de cada módulo é proveniente do *Latch* que aciona um chaveador de potência *(Load Switch)*, habilitando a energização do módulo para o correto funcionamento. O multiplexador possui 4 entradas e 1 saída, trabalhando na faixa de 2.3 a *3.6 VDC*, podendo multiplexar tanto sinais analógicos como digitais. A Figura 3.8 apresenta a Placa Comunicação.

Figura 3.8 Placa Comunicação.

3.5.1 *ZigBee – XBee Pro ZNET 2.5*

ZigBee (ZIGBEE, 2010) é um padrão de comunicação sem fio com baixo consumo de energia. Tal padrão tem se tornado comum para comunicação entre nós de uma RSSF,

portanto foi considerado como o principal módulo para coleta de dados provenientes dos sensores pelo *gateway*.

A empresa *MaxStream* fabrica rádios-transceptores que suportam o protocolo *ZigBee* definido pela *ZigBee Alliance* (ZIGBEE, 2010). Duas versões foram lançadas pela empresa, chamadas de *XBee* e *XBee-PRO* (XBEE, 2010). Ambas transmitem a informação por rádio freqüência, operando na banda de *2.4 GHz*, com taxa de até *250 Kbps*. A Figura 3.9 apresenta o *XBee-PRO ZNET 2.5* da Série 2, ou seja, a segunda geração da família.

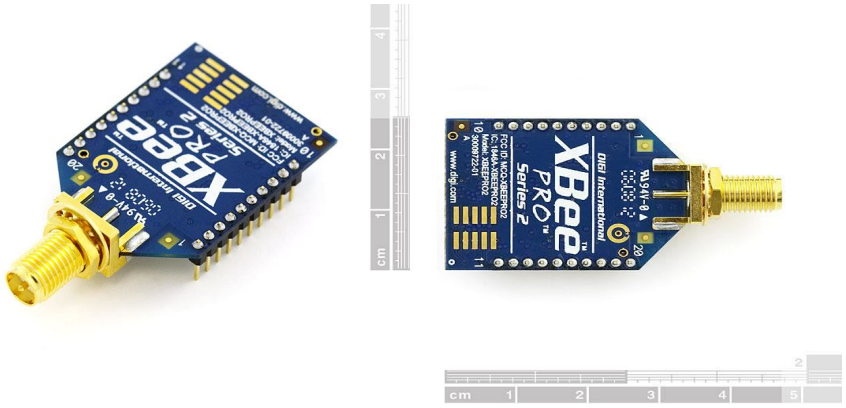


Figura 3.9 *XBee-PRO ZNET 2.5*.

Fonte: Retirado de (XBEE, 2010).

As características de desempenho e potência do módulo *XBee-PRO ZNET 2.5* são apresentadas na tabela 3.1 e as características gerais são apresentadas na tabela 3.2.

Tabela 3.1 – Características de *performance* e potência do módulo *XBee-PRO ZNET 2.5*.

Especificações	***XBee-PRO ZNET 2.5***
	Performance
Alcance (*indoor*)	100 *m*
Alcance (*outdoor*)	1.6 *Km*
Potência Transmitida	63 *mW* (+18 *dBm*) (*Boost Mode* habilitado)
Taxa de Dados (*RF*)	250 *Kbps*
Taxa de Dados (Serial)	1200 – 230400 *bps*

Sensibilidade	-102 *dBm*
Especificações	***XBee-PRO ZNET 2.5***
Potência	
Tensão de Alimentação	3.0 – 3.4 *VDC*
Corrente de Operação (*Tx*)	295 *mA* (3.3 *VDC*)
Corrente de Operação (*Rx*)	45 *mA* (3.3 *VDC*)
Corrente de Operação (*Idle*)	15 *mA*
Corrente (*Power Down*)	<1 μA @ 25ºC

Tabela 3.2 – Características gerais do módulo *XBee-PRO ZNET 2.5*.

Especificações	***XBee-PRO ZNET 2.5***
Gerais	
Frequência de Operação	*ISM 2.4 GHz*
Dimensões	2,438 *cm* x 3,294 *cm*
Temperatura de Operação	-40 a 85ºC
Opções de Antena	Integrada *Whip*, *Chip*, *RPSMA* e *U.FL*
Topologias de Rede	Ponto-a-Ponto; Ponto-Multiponto; *Peer-to-Peer* e *Mesh*
Número de Canais	13 canais direcionais
Endereçamento	*PAN IDs*; e *Clusters IDs*

Cada módulo é constituído por um processador, um bloco transmissor e receptor encapsulados internamente. Com isso, existe a possibilidade de atualização do *firmware* para diversas aplicações, como por exemplo, a seleção do modo de comunicação *AT* ou *API*, a modificação de registradores internos para endereçamento, criptografia, taxa de transmissão, potência de saída entre outros.

No modo de comunicação *AT*, cada módulo transmite os dados da mesma forma que em uma comunicação serial *RS-232* padrão. Os módulos dispõem de *buffers* de transmissão e recepção para uma melhor desempenho na comunicação serial.

No modo *API* os dados transmitidos e recebidos estão contidos em *frames*, que definem operações ou eventos dentro do módulo. Através desse modo de operação é possível um determinado módulo enviar endereço fonte, endereço destino, nome de um determinado nó, sinal *RSSI*, estado, etc.

O diagrama em bloco da constituição interna do módulo *XBee-PRO ZNET 2.5* é apresentado na Figura 3.10.

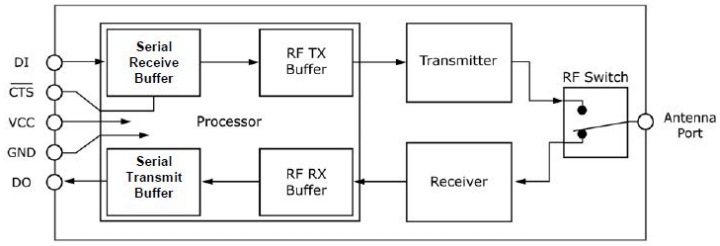


Figura 3.10 Diagrama em Blocos do *XBee-PRO ZNET 2.5*.

Fonte: Retirado de (XBEE, 2010).

O *XBee-PRO ZNET 2.5* pode operar tanto com o protocolo *ZigBee* como apenas sob o protocolo *IEEE 802.15.4*.

O protocolo *IEEE 802.15.4* permite a criação de redes ponto-a-ponto, ponto-multiponto e *peer-to-peer*. Embora o protocolo *IEEE 802.15.4* necessite da figura de um coordenador na rede, os módulos da *MaxStream* podem ser utilizados sem que esta opção esteja ativada (SCHWARZ, 2007).

O protocolo *ZigBee* é baseado no *IEEE 802.15.4*, com as vantagens de permitir a implementação de topologias de rede *Mesh* e a capacidade de auto-procura. Em redes pequenas, a topologia *Mesh* e a capacidade de auto-procura não são necessárias, portanto, o protocolo utilizado pode ser simplesmente o *IEEE 802.15.4* (SCHWARZ, 2007).

Segundo SCHWARZ (2007), os módulos permitem ainda duas formas de endereçamento dos pacotes para se ter um mínimo de segurança na rede: endereçamento em 16 *Bits* ou endereçamento em 64 *Bits*. O modo de endereçamento de 16 *Bits* requer que o endereço seja gravado no *firmware* do rádio. Este endereço deve ser escolhido de forma a permanecer único dentro da rede. Cada módulo *XBee* possui um número de 64 *Bits* fornecido pelo *IEEE* que é único em escala mundial.

3.5.2 *Bluetooth – BlueSmirf*

Dispositivos móveis como celulares e *notebooks* dispõem normalmente de *interfaces Bluetooth* (BLUETOOTH, 2010). Pode-se considerar que tais dispositivos possam se comunicar com uma RSSF através do *gateway*. Dessa forma, optou-se por introduzir esse módulo na implementação inicial.

O *BlueSmirf* (SPARKFUN, 2010) é um módulo de comunicação que utiliza a tecnologia *Bluetooth* para implementar um canal de comunicação *full-duplex* entre um microcontrolador e qualquer outro dispositivo que tenha uma *interface* de rede *Bluetooth.* As principais características técnicas do *BlueSmirf* são apresentadas na Tabela 3.3.

Tabela 3.3 – Características técnicas do módulo *BlueSmirf*.

Características	*BlueSmirf*
Frequência de Operação	2.4 *GHz ISM*
Modulação	*GFSK (Gaussian Frequency Shift Key)*
Alcance	Até 10 *m*
Taxas de Transmissão	2400 – 115200 *bps*
Largura de Banda	200 *Kbps* (*fast mode*) 60 *Kbps* (*regular data mode*)
Potência de Transmissão	- 9 *dBm* ~ 15 *dBm* (Configurável)
Tensão de Alimentação	3 – 6 *VDC*

Consumo de Corrente (conexão)	48 *mA*
Consumo de Corrente (normal)	33 *mA*
Consumo de Corrente (*Stand by*)	3 *mA*
Latência de Dados	2 ~ 20 *ms*
Dimensões Físicas	43 *mm* x 15 *mm*
Peso	2 *g*

Na prática, o *BlueSmirf* é um modem que utiliza como *interface* de comunicação o padrão *UART (RS-232)* e foi concebido com base em um núcleo *SoC*. Esse núcleo implementa grande parte da pilha de protocolos *Bluetooth* no próprio *hardware* (BARBOSA, 2008). A Figura 3.11 apresenta o diagrama em blocos do módulo.

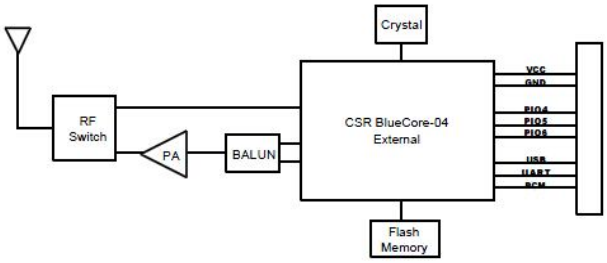


Figura 3.11 Diagrama em Blocos *do BlueSmirf*.

FONTE: Retirado de (SPARKFUN, 2010).

Segundo (BARBOSA, 2008) a pilha de protocolos *Bluetooth* inclui serviços como autenticação, nomeação, descoberta e roteamento. O principal benefício dessa implementação é a liberação de recursos do microcontrolador. A Figura 3.12 apresenta o *BlueSmirf*.

Figura 3.12 *BlueSmirf*.

FONTE: Retirado de (SPARKFUN, 2010).

Para gerência do dispositivo, o *BlueSmirf* disponibiliza um conjunto de comandos, com sintaxe muito parecida com os comandos *Hayes*. Isso facilita a manutenção e o monitoramento do canal de comunicação (BARBOSA, 2008).

3.5.3 *GSM/GPRS – Telit GM862*

Uma forma viável e bastante comum de permitir a coleta de dados armazenados em um *gateway* é através de uma conexão *GSM/GPRS* pela rede de telefonia celular. Dessa forma, um módulo com tal tecnologia também é empregado como opção inicial.

Segundo (TELIT, 2009), o módulo GM862, possui funcionalidades que podem ser aplicáveis a sistemas de segurança, telemetria e controle de sistemas utilizando rede celular. O módulo possui capacidade de comunicação via *GSM/GPRS* e compatibilidade com protocolo *TCP/IP*. A Figura 3.13 apresenta o módulo *Telit GM862*.

Figura 3.13 *Telit GM862*.

FONTE: Retirado de (TELIT, 2009).

A *interface* de comunicação e configuração do módulo *Telit GM862* é o padrão *UART (RS-232)*, o mesmo necessita de um conector *Molex* de 50 pinos e um nível de tensão de *3.8 V* para alimentação e de *3.3V TTL* para comunicação para o correto funcionamento. O módulo pode ser configurado através de comandos *AT*, podendo enviar mensagens *SMS*, *E-mail*,

transferência de dados pela rede *GPRS* e até mesmo criar chamadas telefônicas pela rede *GSM*.

3.6 Unidade Sensoriamento

A Unidade Sensoriamento (esquemático apresentado no Apêndice A.5) foi desenvolvida com o intuito de proporcionar à arquitetura a condição de monitoramento do ambiente através de sensores. A unidade possui um sensor de temperatura *LM35* e um de luminosidade *LDR* (***L**ight **D**ependent **R**esistor* ou em português *Resistor Dependente de Luz).* O diagrama em blocos da unidade é apresentado na Figura 3.14.

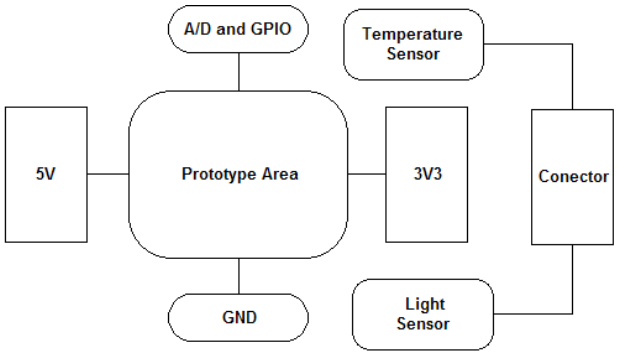


Figura 3.14 Diagrama em Blocos da Unidade Sensoriamento.

O sensor de temperatura escolhido é um *LM35*, por ser barato, preciso e ter uma ampla faixa linear de medição que vai de *-55°C* a *150°C*. Este sensor informa a temperatura por meio de uma tensão em *volts*. Se a referência for o terra, então *23.5°C* será apresentada na saída como *0.235V*. Cada incremento de *0.1°C* representa um incremento de *1 mV* na tensão.

O *LDR* é um transdutor de entrada (sensor) que converte a (luz) em valores de resistência. É feito de sulfeto de cádmio (*CdS*) ou seleneto de cádmio (*CdSe*). Sua resistência diminui quando a luz é muito alta, e quando a luz é baixa, a resistência no *LDR* aumenta. Um multímetro pode ser usado para encontrar a resistência na escuridão ou na presença de luz intensa. Estes são os resultados típicos para um *LDR* padrão:

• Escuridão : resistência máxima, geralmente acima de *1M ohms*.

• Luz muito brilhante : resistência mínima, aproximadamente *100 ohms*.

Uma área de prototipação foi desenvolvida formada por 16 colunas de pontos interconectados, formando uma grande matriz de contatos para possíveis montagens de circuitos condicionadores de sinais e diversos outros sensores que possam ser utilizados. A Figura 3.15 apresenta a Placa Unidade Sensoriamento.

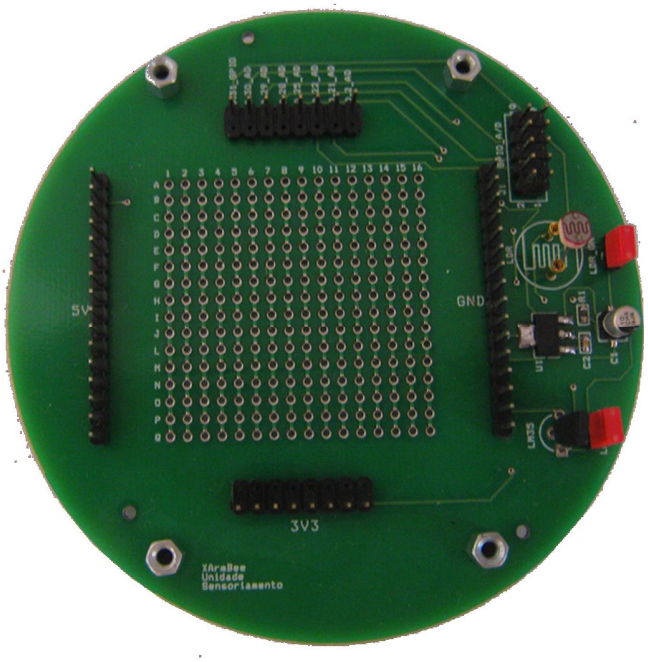


Figura 3.15 Placa Unidade Sensoriamento.

3.7 *XBeeNode*

A placa *XBeeNode* (esquemático apresentado no Apêndice A.6) foi desenvolvida para ser utilizada na atualização do *firmware* do módulo *ZigBee* (*XBee-PRO*), complementar à arquitetura do *gateway* e para atuar como nó sensor na RSSF experimental para validar as funcionalidades da arquitetura. O Placa *XBeeNode* é constituída basicamente de um circuito conversor *USB/Serial*, dois sensores sendo um *LM35* (Temperatura) e um *LDR* (Luminosidade). A placa possui um regulador de tensão de *3V3* para alimentação do módulo

XBee-PRO e um soquete para quatro pilhas do tipo *AAA*. A Figura 3.16 apresenta o diagrama em blocos da placa *XBeeNode*.

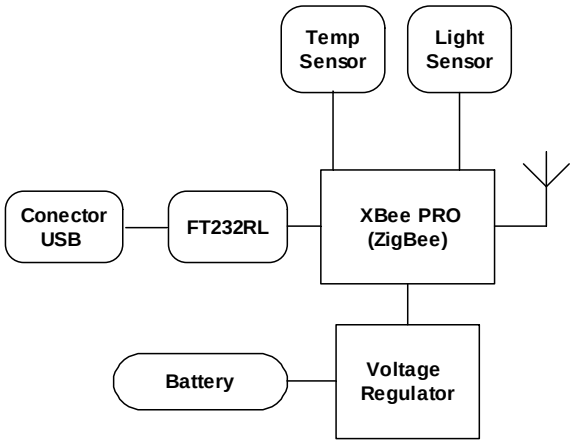


Figura 3.16 Diagrama em Blocos da Placa *XBeeNode*.

A placa *XBeeNode* possui também conectores de expansão para *interface* com um microcontrolador, 3 pinos *ADCs* do próprio módulo *XBee-PRO* e conectores de *5V* e *GND* para alimentação de circuitos externos. A Figura 3.17 apresenta a placa *XBeeNode*.

Figura 3.17 Placa *XBeeNode*.

3.8 Ferramentas de *Software*

Com o objetivo de configuração e desenvolvimento de *software* do sistema proposto, são apresentadas as ferramentas utilizadas para o desenvolvimento das funções (*WinARM*),

para o carregamento do *firmware* no microcontrolador (*Flash Utility*) e configuração dos módulos *ZigBee* (*X-CTU*).

3.8.1 *WinARM*

O *software WinARM* (WINARM, 2006) é um ambiente de desenvolvimento para microcontroladores *ARM7*, de fácil utilização e que faz uso do compilador *GNU-C/C++-Compiler Version 4.1.1*, gratuito com suporte a várias linguagens de programação como *C, Java, C#, VHDL, Assembly* entre outras (MARTINIANO JÚNIOR, 2008) . Portanto, pode-se desenvolver aplicações tanto em linguagem *C/C++* quanto em *Assembly*, ficando a critério do projetista a escolha mais adequada.

O *software* possui uma tela de visualização do resultado da execução do programa onde se pode observar o processo de compilação do código fonte. No final da compilação o ambiente gera um arquivo com extensão *.hex* que é o *firmware* que será embarcado no microcontrolado *ARM7*.

Na criação de um novo projeto o *WinARM* já possui o *Makefile*, que é o código de mapeamento do microcontrolador e da seleção do modo de gravação que será utilizado para carregar os arquivos *.hex* desenvolvidos. Existem alguns exemplos de aplicações com diversos microcontroladores da família *ARM7, CORTEX* entre outros. A Figura 3.18 apresenta a edição de um código escrito em linguagem *C*.

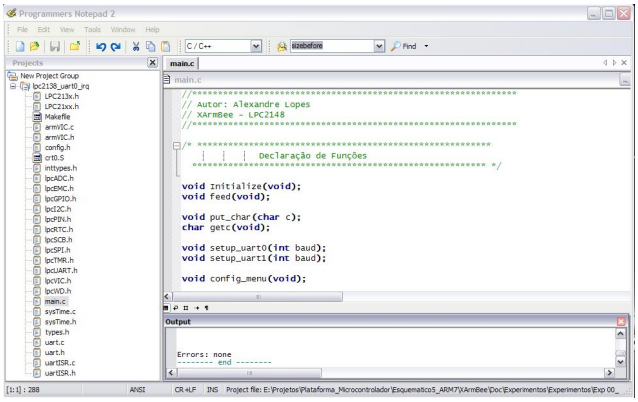


Figura 3.18 *Software WinARM*.

3.8.2 *LPC2000 Flash Utility V 2.2.3*

O *software Flash Utility* (FLASH UTILITY, 2004) é utilizado para gravar o *firmware* desenvolvido no microcontrolador *ARM7* da Placa Processamento e Armazenamento de Dados. Para que a gravação seja realizada com sucesso, é necessário que as seguintes configurações no *Flash Utility* sejam realizadas, conforme descrito abaixo:

A primeira parte da configuração do *software Flash Utility* consiste em definir a porta serial que se deseja utilizar. Em seguida é definida a taxa de velocidade da porta de comunicação serial, que pode variar entre 110 a 115200 *bps*, a seleção do tempo de espera do sistema e a habilitação do pino *DTR/RTS* para o uso de *resets* durante a gravação.

A segunda parte da configuração do *Flash Utility* consiste em identificar o microcontrolador e a frequência de operação do oscilador em que o microcontrolador irá trabalhar. Estas configurações são feitas automaticamente, ao se clicar no botão *Read Device ID* (Ler Identificação do Dispositivo).

Após identificar o microcontrolador, é preciso preparar a memória *Flash* para receber os dados que serão gravados. Para isso, utiliza-se o botão *Blank Check* (Checar Memória) para verificar se a memória está limpa e em seguida o botão *Erase* (Apagar) para garantir que a memória seja apagada. A Figura 3.19 apresenta todos os passos citados acima.

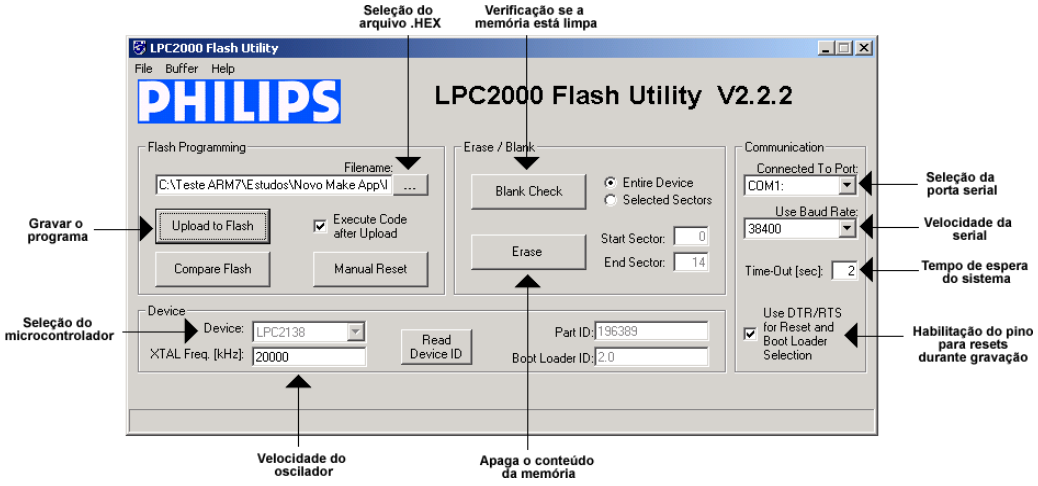


Figura 3.19 *Software Flash Utility V2.2.3*

FONTE: Retirado de (FLASH UTILITY, 2004).

Por fim, seleciona-se o arquivo *.hex* que se deseja gravar no microcontrolador, por meio da opção *Filename* (Nome do Arquivo), e pressiona-se o botão *Up Load to Flash* (Gravar para Memória) para finalizar o processo de gravação.

3.8.3 *X-CTU*

O *software X-CTU* (X-CTU, 2007) é utilizado para configurar os módulos *XBee-PRO* e para testar a comunicação entre eles, ou seja, verificar se há a transmissão e o recebimento de dados entre os módulos. Ao conectar o(s) módulo(s) *XBee-PRO* no computador e executar o programa *X-CTU* o mesmo verifica e reconhece o(s) módulo(s) conectado(s) na porta *USB* conforme ilustrado na Figura 3.20, possibilitando através de uma *interface* gráfica, a configuração dos pinos e a programação do módulo por meio da opção *terminal*.

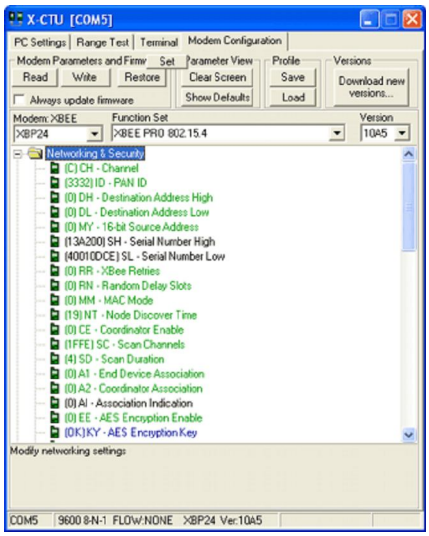


Figura 3.20 *Software X-CTU*.

FONTE: retirado de (X-CTU, 2007).

Por meio do *X-CTU*, pode-se ter acesso a todas as funcionalidades e recursos disponíveis do módulo *XBee-PRO* uma vez que o *firmware* carregado foi selecionado para a aplicação específica desejada, ou seja, o funcionamento como roteador, coordenador ou dispositivo final em uma RSSF. É possível alterar os parâmetros dos registradores do módulo para a aplicação, ou seja, configuração do módulo na rede.

É possível enviar comandos pelo emulador de terminal a outro módulo *XBee-PRO* e verificar a funcionalidade dos mesmos, semelhante ao *Hyper Terminal* do sistema operacional *Windows*.

3.9 Arquitetura de *Software*

A arquitetura de *software* é formada por funções básicas de ativação, multiplexação, comunicação e armazenamento de dados. Todas essas funções foram desenvolvidas e formam uma *API* com os exemplos utilizados neste projeto.

Um fluxograma geral é apresentado na Figura 3.23 contendo o relacionamento entre as funções e o princípio de funcionamento da arquitetura.

3.9.1 *XArmBee – API*

A *XArmBee – API* é basicamente formada por um conjunto de funções de sinalização, ativação, multiplexação, armazenamento de dados e comunicação entre os módulos utilizados na arquitetura.

As funções de sinalização, ativação e multiplexação são baseadas respectivamente pela tabela da verdade de circuitos integrados. As funções de sinalização e ativação dos módulos dependem do *Latch tipo D (74HCT573)*. A função de multiplexação depende de uma chave analógica *TS3A5017* que seleciona a saída em função da combinação lógica existente com os níveis de entrada.

O circuito de habilitação (ativação) dos módulos de comunicação da arquitetura é apresentado na Figura 3.21. Analisando o circuito verifica-se que o sinal de entrada é habilitado na saída, somente se o nível de *LE* estiver em nível lógico alto (1). Os *leds* D17 (*ZigBee*); D16 (*Bluetooth*) e D15 (*GSM/GPRS*) sinalizam quando cada módulo está ativado ou desativado.

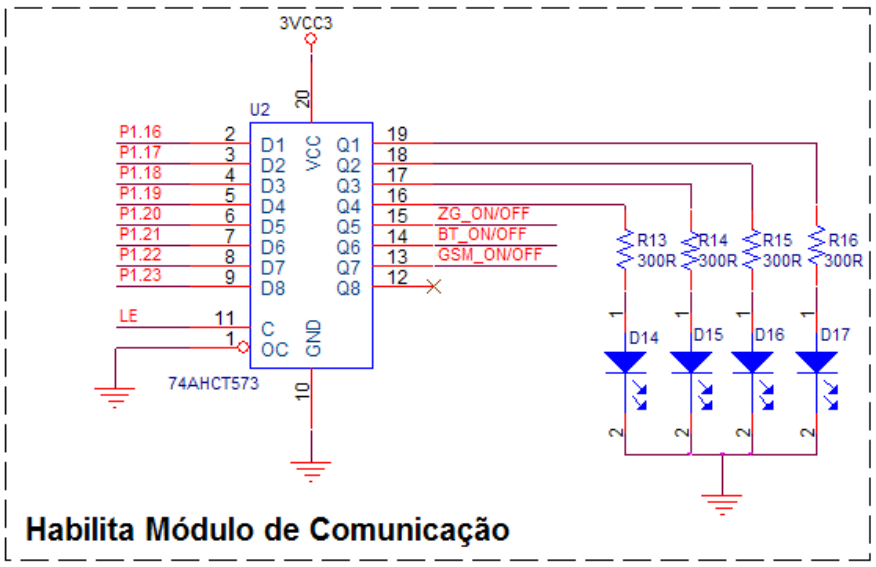


Figura 3.21 Circuito para habilitação dos módulos de comunicação.

Os módulos são ativados com os sinais dos pinos P1.20 (*ZigBee*); P1.21 (*Bluetooth*) e P1.22 (*GSM/GPRS*). O mesmo princípio é utilizado para multiplexação dos sinais da *UART 1* do microcontrolador para comunicação com os módulos. A Figura 3.22 apresenta o circuito de multiplexação dos módulos de comunicação.

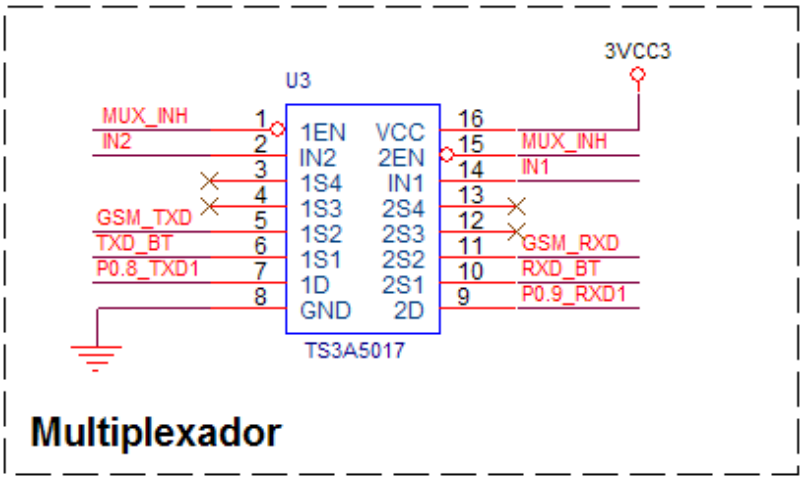


Figura 3.22 Circuito para multiplexação dos módulos de comunicação.

O sinal *MUX_INH* deve sempre estar em nível lógico baixo e os sinais de *IN1* e *IN2* devem seguir a seguinte combinação: 00 – Aciona o módulo *Bluetooth* e 01 – Aciona o módulo *GSM/GPRS* respectivamente.

Partindo desse princípio a tabela 3.4 foi montada com a combinação correta para a sinalização, ativação e multiplexação de cada módulo de comunicação.

Tabela 3.4 – Pinos de habilitação dos módulos e pinos de controle de sinalização.

x	Pinos do Microcontrolador	*ZigBee/Bluetooth*	*ZigBee/GPRS*
0	P1.16 – D17	1	1
1	P1.17 – D16	1	0
2	P1.18 – D15	0	1
3	P1.20 – ZG	1	1
4	P1.21 – BT	1	0
5	P1.22 – GSM/GPRS	0	1
6	P1.26 – LE	1	1
7	P1.27 – MUX_INH	0	0
8	P1.28 – IN1	0	0
9	P1.29 – IN2	0	1

A função em *C* que implementa a sinalização, ativação e multiplexação de cada módulo de acordo com a tabela da verdade acima é a seguinte:

```
/*********************************************************
     Função de Sinalização - Ativação - Multiplexação
*********************************************************/


void liga16_17_20_21_26(void)  // ZigBee e Bluetooth
{

  PINSEL1  = 0x00000000; //Configura os pinos da porta 1 como GPIO
  IO1DIR   = 0x34FF0000; //Configura os Pinos P1.26 e de P1.16 a P1.23 como Saída
  IO1SET   = 0x04330000; // Seta os pinos P1.26, P1.16, P.17, P1.20 e P1.21 alto
  IO1CLR   = 0x0BCC0000; // Seta os demais em baixo

}

void liga16_18_20_22_26_29(void)  // ZigBee e GSM/GPRS
{

  PINSEL1  = 0x00000000;
  IO1DIR   = 0x34FF0000;
  IO1SET   = 0x24550000;
  IO1CLR   = 0xDBAA0000;
}
```

3.9.2 Modelo de Aplicação

Com as *APIs* apresentadas, pode-se construir aplicações conforme as necessidades de um projeto específico. Esta seção apresenta um modelo específico que exemplifica e pode nortear novos desenvolvimentos. A Figura 3.23 apresenta o fluxograma de um modelo de aplicação para a arquitetura. O código da aplicação é apresentado o Apêndice B.

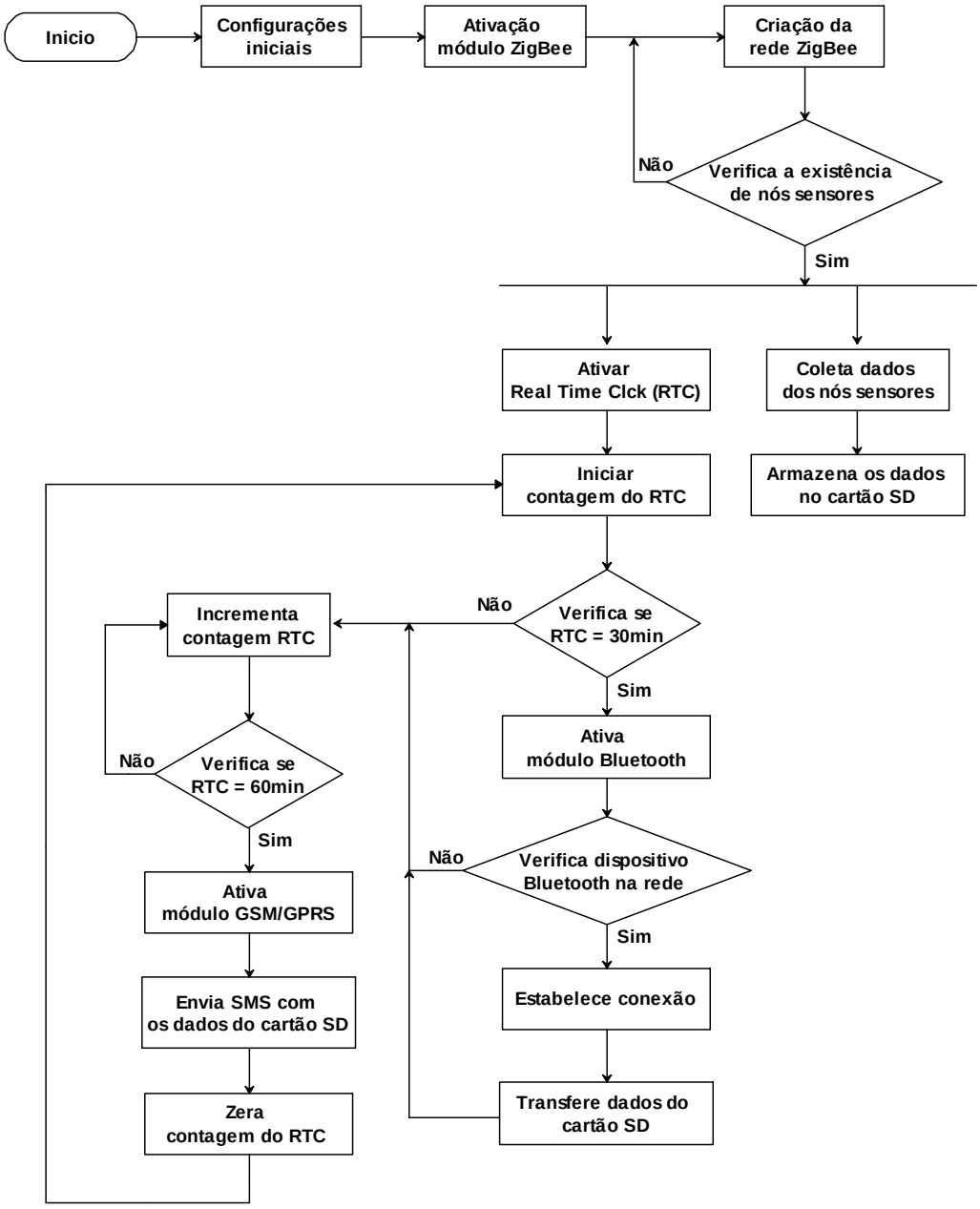


Figura 3.23 Fluxograma de uma aplicação básica da arquitetura *XArmBee.*

Analisando o fluxograma apresentado podemos verificar que o funcionamento da arquitetura *XArmBee* em uma aplicação típica consiste na execução dos passos a seguir:

- A configuração inicial do microcontrolador através dos pinos que serão utilizados como entrada e saída e as *interfaces SPI*, Serial etc.
- Após a definição dos pinos é ativado o módulo *ZigBee* para que seja inicializado seu modo de operação e com isso cria-se uma rede para detecção de outros módulos presentes caso estejam habilitados.
- Verificado a existência de nós sensores habilitados na rede, inicia-se então a coleta de dados dos nós sensores juntamente com o armazenamento dos pacotes de dados em um cartão *SD* de *2 GB*. Em paralelo a isso, é feita a ativação do *Real Time Clock (RTC)* e a inicialização de sua contagem.
- A cada 30 minutos o *RTC* realiza a ativação do módulo *Bluetooth* que verifica se há algum dispositivo solicitando conexão na rede e caso exista estabelece conexão para que seja feita a transferência dos dados armazenados no cartão *SD*.
- Ao final da transferência ou se nenhum dispositivo *Bluetooth* for encontrado na rede a contagem do *RTC* é incrementada.
- A cada 60 minutos do *RTC* é ativado o módulo *GSM/GPRS* que é responsável pelo envio de mensagens *SMS* contendo dados armazenados no cartão *SD* para outro dispositivo da rede celular.
- No final do envio das mensagens o procedimento de contagem do *RTC* é zerado e reinicializado para repetição do fluxo de execução do processo de coleta, armazenamento, comunicação e transmissão de dados realizado pela arquitetura.

4 Testes e Avaliação

Neste capítulo são apresentados testes para validação individual de cada módulo de comunicação e avaliação do consumo de corrente da arquitetura com relação à ativação e multiplexação dos módulos, também é apresentada uma avaliação qualitativa da arquitetura, em comparação com outras existentes e são descritas algumas execuções de estudos de caso.

4.1 Módulos de Comunicação

Os módulos utilizados para realização dos testes e validação inicial foram: *XBee-PRO ZNET 2.5* (*ZigBee*); *BlueSmirf* (*Bluetooth*) e o *GM862* (*GSM/GPRS*). Todos os módulos permitem a comunicação pela serial através de comandos *AT*. Dessa forma facilitando a *interface* com o microcontrolador *LPC2148* (*ARM7).*

4.1.1 Comunicação *ZigBee*

Para validação da arquitetura realizando uma comunicação *ZigBee*, foi utilizada uma Placa *XBeeNode* conectada a um *PC* para recebimento dos pacotes de dados enviados da arquitetura *XArmBee*. A Figura 4.1 apresenta uma ilustração da comunicação *ZigBee*.

Figura 4.1 Comunicação *ZigBee*.

O procedimento para realização do teste será descrito através dos seguintes passos:

1. O primeiro passo é a configuração do módulo *XBee-PRO ZNET 2.5* que ficará na arquitetura. Utilizando-se a placa *XBeeNode* para gravação do *firmware* e o *software X-CTU* para configurar os registradores na Versão: *XBP24-ZB ZNET 2.5 COORDINATOR API.* A Figura 4.2 apresenta a configuração dos registradores para o módulo.

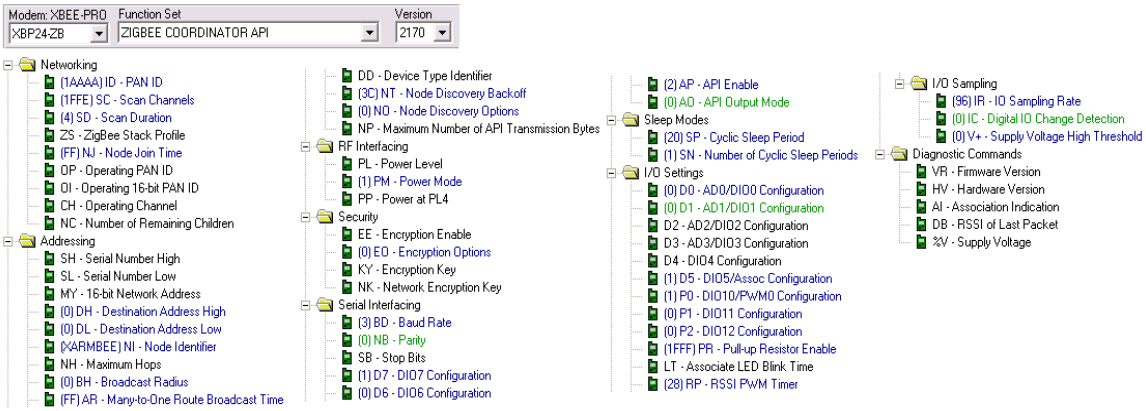


Figura 4.2 Registradores do módulo *XBee-PRO* da arquitetura *XArmBee.*

A Figura 4.3 apresenta o resultado final após o processo de gravação e atualização do *firmware* do módulo *XBee-PRO* da arquitetura.

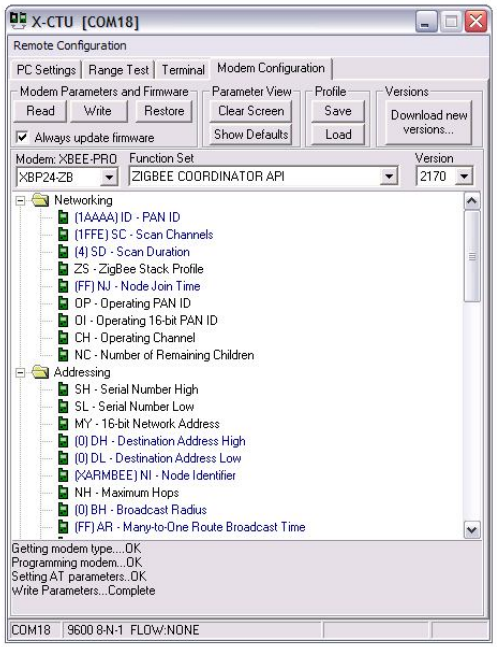


Figura 4.3 Atualização do módulo *XBee-PRO* da arquitetura *XArmBee.*

2. O segundo passo é a configuração do módulo que ficará na placa *XBeeNode* com o *firmware* na Versão: *XBP24-B ZNET 2.5 ROUTER/END DEVICE API*. A Figura 4.4 apresenta a configuração dos registradores para o módulo.

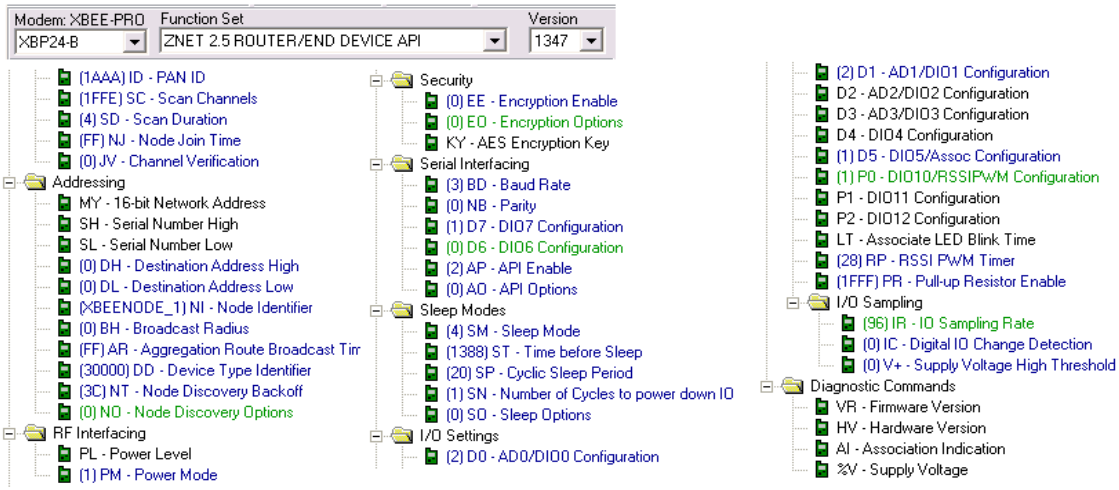


Figura 4.4 Registradores do módulo *XBee-PRO* do *XBeeNode*.

A Figura 4.5 apresenta o resultado final após o processo de gravação e atualização do *firmware do* módulo *XBee-PRO* do *XBeeNode*.

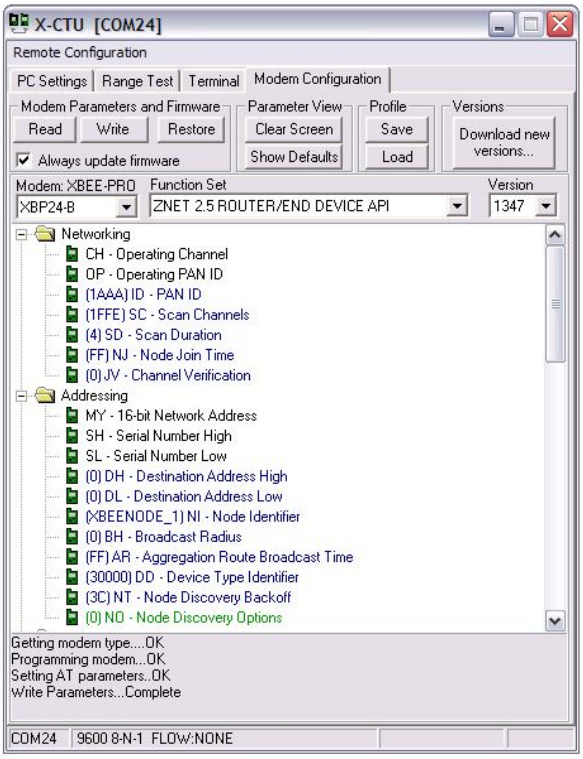


Figura 4.5 Atualização do módulo *XBee-PRO* do *XBeeNode*.

3. O último passo é abrir o *Hyper Terminal* ou *software* equivalente e realizar a seguinte configuração: *Bits* por segundo: 9600; *Bits* de Dados: 8; Paridade: Nenhum; *Bits* de parada: 1; Controle de Fluxo: Nenhum.

A Figura 4.6 apresenta os resultados obtidos no *Hyper Terminal*, mostrando os pacotes de dados recebidos do *XBee-PRO* no modo *API*.

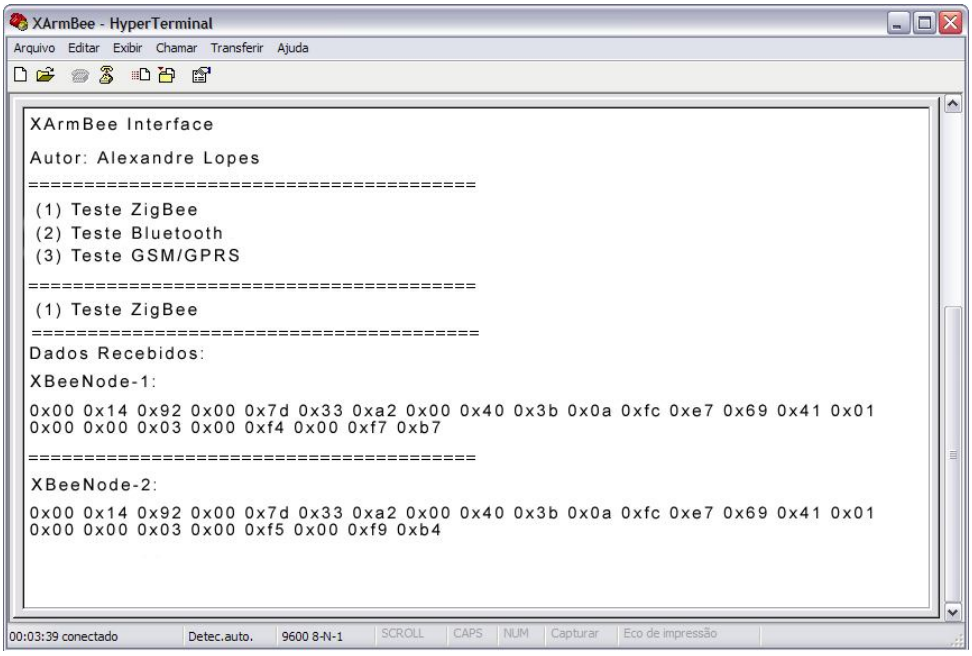


Figura 4.6 Resultados da comunicação *ZigBee no Hyper Terminal*..

No modo *API* o módulo *XBee-PRO* envia o pacote de dados no formato de *frames* que possuem uma estrutura rígida e bem definida pelo fabricante, tornando-se uma opção de comunicação interessante, pois possibilita a criação de estruturas de redes mais complexas e até mesmo a configuração remota de membros da rede.

Os dados apresentados na Figura 4.6 representam os *frames* foram previamente coletados dos sensores de temperatura e luminosidade de outras duas placas *XBeeNode* e correspondem aos valores em hexadecimal lidos e armazenados no cartão *SD* de *2 GB* existente na arquitetura. O Apêndice B apresenta as rotinas de ativação e comunicação utilizadas para validação do teste *ZigBee* juntamente com as rotinas para o cartão *SD* utilizadas para armazenamento de dados.

4.1.2 Comunicação *Bluetooth*

A comunicação *Bluetooth* consiste em transmitir os mesmos dados armazenados no cartão *SD* de *2 GB* do experimento com o módulo *XBee-PRO* com a diferença da mudança de tecnologia. Para validar o experimento realizou-se uma conexão serial utilizando-se *Bluetooth* com um *dongle* conectado a um *PC*. A Figura 4.7 apresenta o cenário utilizado para o experimento.

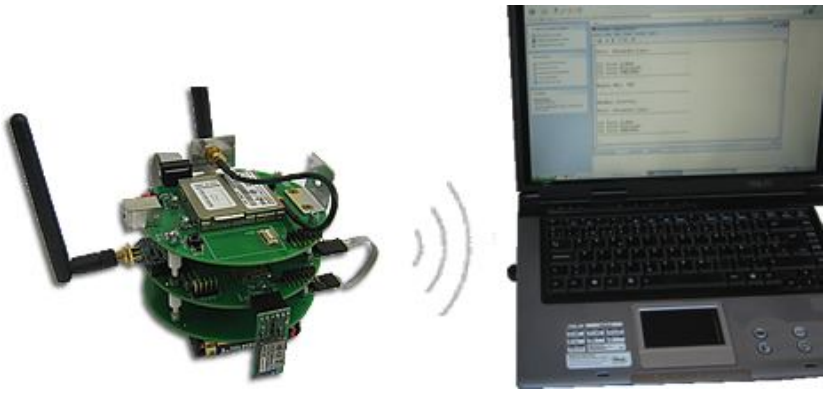


Figura 4.7 Comunicação *Bluetooth*.

As função de ativação e configuração do módulo *BlueSmirf* são apresentadas no Apêndice B. Para recebimento dos dados basta configurar o *Hyper Terminal* do mesmo modo realizado para a comunicação *ZigBee*. Os resultados obtidos são apresentados conforme Figura 4.8.

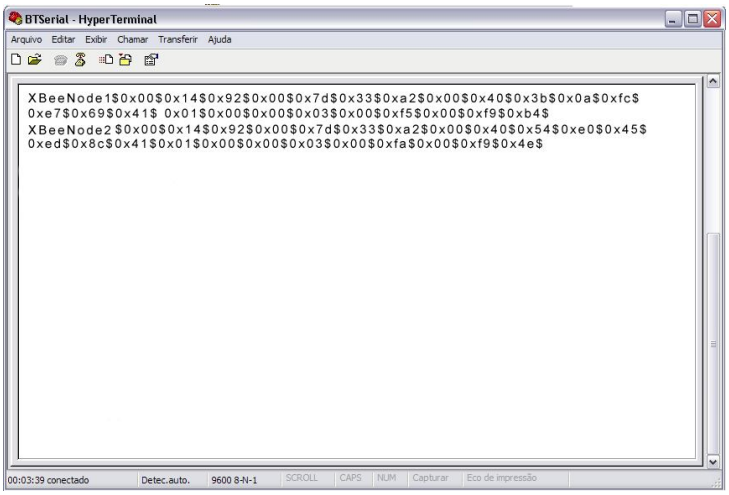


Figura 4.8 Resultados da comunicação *Bluetooth no Hyper Terminal*

4.1.3 Comunicação *GSM/GPRS*

O experimento da comunicação *GSM/GPRS* consistiu em enviar uma mensagem *SMS* para um número previamente determinado com os valores armazenados no cartão *SD* de *2GB*. A Figura 4.9 apresenta o cenário montado para o experimento.

Figura 4.9 Comunicação *GSM/GPRS*

O algoritmo que utiliza um conjunto de comandos *AT* para ativação, armazenamento e envio da mensagem *SMS* utilizando o módulo *Telit GM862* é apresentado abaixo.

****

Algoritmo para enviar uma mensagem SMS Salvando o Número:

1. AT // Enter para testar se o modem estar respondendo

2. Selecionar o tipo do formato da Mensagem SMS:

    AT+CMGF =1 // Tipo Texto

    Response = OK

3. AT+CMGW="9281329584" // Número do Telefone que se quer enviar a mensagem

    > Mensagem.... // Digitar a Mensagem depois de > e no final *ctrl -z*

    +CMGW: 1 // Ficou armazenada a mensagem na posição 1

    OK

4. AT+CMSS=1 // Comando para enviar a mensagem armazenada na posição 1

****

4.2 Medição do Consumo de Energia da Arquitetura

A principal característica da arquitetura *XArmBee* é o suporte a múltiplas *interfaces* de comunicação sem fio, sendo expansível a outras tecnologias que tenham o padrão de comunicação feito através de uma *interface UART(RS-232)*. A arquitetura tem como princípio de funcionamento a gerência do consumo de energia por módulo, ou seja, o módulo é ativado somente quando selecionado e em determinados instantes de tempo, proporcionando uma maior autonomia à arquitetura.

Para validação desta funcionalidade realizou-se os experimentos de medição do consumo de corrente de acordo com o chaveamento realizado entre os módulos de comunicação. Todos transmitindo um pacote de dados fixo de 10 *KB*. A Figura 4.10 apresenta o cenário proposto.

Os procedimentos realizados para medição do consumo de corrente e ativação de cada módulo de acordo com a aplicação foram os seguintes:

(a) Medição da tensão sobre o resistor *shunt* através de um multímetro de precisão, quando somente o módulo *XBee-PRO ZNET 2.5(ZigBee)* está acionado.

A medição da tensão V_{SHUNT} (V_O) resultou em um nível de tensão de 0,129 *VDC* que aplicado na equação abaixo, com os valores de $R_L = 5\ K\Omega$ *e* $R_S = 1{,}5\ \Omega$, tem-se o seguinte valor de I_S:

$$V_0 = \frac{I_S x\ R_S x\ R_L}{5k}$$

$$I_S = \frac{0{,}129\ x\ 5k}{1{,}5\ x\ 5k} = 0{,}086A$$

O valor de I_S corresponde ao acionamento somente do módulo *ZigBee*.

A medição dos valores de corrente para os outros módulos casos: (b) e (c) da Figura 4.10 seguem o mesmo procedimento tendo como princípio a habilitação do módulo e o chaveamento da *interface* de comunicação. Para validação utiliza-se um código simples para teste da funcionalidade.

A arquitetura possui quatro *leds* que estão conectados ao *Latch Enable* com isso podemos visualizar quando um módulo está ou não ativo. Observamos a mudança de acionamento entre módulos.

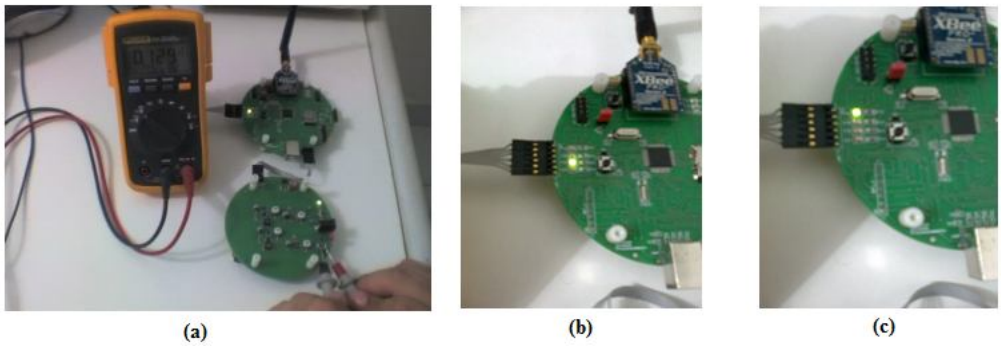


Figura 4.10 Consumo de corrente e chaveamento entre os módulos de comunicação: (a) *ZigBee*; (b) *Bluetooth*; (c) *GSM/GPRS*.

A Tabela 4.1 apresenta os valores de corrente medidos para cada módulo de comunicação do experimento realizado.

Tabela 4.1 – Valores da corrente I_S medidos no experimento.

Módulo de Comunicação	*XBee-PRO ZNET 2.5*	*BlueSmirf*	*Telit GM862*
Corrente I_S (*A*)	0,086	0,065	0,105

4.3 Avaliação Qualitativa

Avaliar um sistema é pronunciar-se sobre as características do mesmo, ou seja, é realizar toda e qualquer observação sobre o mesmo (PIDD, 2005). Com isso existem basicamente dois tipos de avaliação, qualitativa e quantitativa. Na qualitativa, existe a necessidade de uma operação com o senso comum ou com valores de referência. A avaliação quantitativa baseia-se na formulação de valores específicos sem levar em consideração os méritos dos valores obtidos. Baseado nesse princípio realizou-se uma avaliação qualitativa da

arquitetura *XArmBee* em relação a outras arquiteturas de *hardware* existentes, verificando em quais aspectos a arquitetura possui pontos fortes em relação as demais.

A Tabela 4.2 demonstra o comparativo realizado entre o *XArmBee* e outras arquiteturas de *hardware*.

Tabela 4.2 – Comparativo entre *XArmBee* e outras arquiteturas.

Arquitetura de *Hardware*	***BTnode***	***Mica2***	***Mica2Dot***	***Tmote Sky***	***Imote***	***XArmBee***
Microcontrolador	*ATMega128L*	*ATMega128L*	*ATMega128L*	*MSP430F*	*ARM7*	*ARM7*
Arquitetura	8 *Bits*	8 *Bits*	8 *Bits*	16 *Bits*	32 *Bits*	32 *Bits*
Velocidade	7.37 *MHz*	7.37 *MHz*	4 *MHz*	8 *MHz*	12 *MHz*	12 *MHz*
Memória de Programa	128 *KB*	128 *KB*	128 *KB*	48 *KB*	512 *KB*	512 *KB*
Memória de Dados	64 *KB*	4 *KB*	4 *KB*	10 *KB*	11 *KB*	32 *KB*
Memória de Armazenamento	180 *KB*	512 *KB*	512 *KB*	1024 *KB*	---------	2 *GB*
I/O Externos	40	51	18	16	30	45

Analisando a Tabela 4.2 verifica-se o grande potencial de processamento e armazenamento de dados da arquitetura *XArmBee* com a possibilidade de expansão das conexões de (*I/O*).

A Figura 4.11 apresenta uma outra forma de apresentação dos dados através de gráficos tipo estrela multidimensionais.

Foram construídos quatro gráficos, onde cada um possui uma característica a ser analisada como, por exemplo:

(a) **Arquitetura:** compara a arquitetura *XArmBee* em relação a quantidade de *bits* de processamento.

(b) **Velocidade de Processament**o: comparação em relação a frequência de operação da arquitetura em (MHz).

(c) Memória de Programa: comparação em relação a memória de programa em (KB).

(d) **Possíveis *Interface* de Rádio:** Quantidade possível de rádios-transceptores suportados pela arquitetura.

Todas as arquiteturas são visualizadas simultaneamente no mesmo gráfico, com isso pode-se observar de forma clara qual a tendência, ou seja, qual característica da arquitetura deve ser explorada de forma mais eficiente.

Na fase inicial de concepção e escolha de uma arquitetura para um projeto específico, torna-se um ferramenta fundamental para a correta seleção.

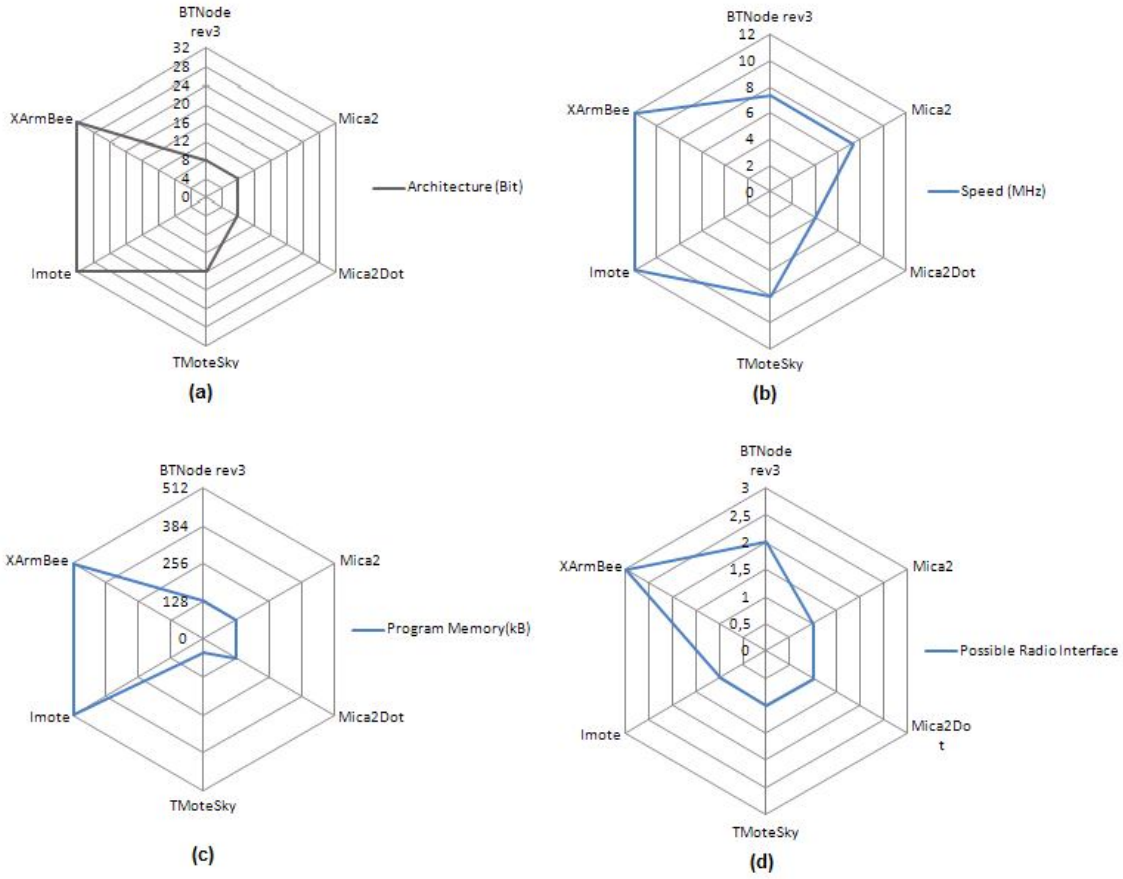


Figura 4.11 Comparação das arquiteturas de *hardware*.

4.4 Estudo de Caso

Será apresentado um estudo de caso para uma aplicação de coleta presencial de dados através de uma rede *Bluetooth* e outro para coleta remota de dados através de uma rede de dados *GSM/GPRS* utilizando a arquitetura *XArmBee*.

O intuito principal é apresentar uma aplicação prática para validar a funcionalidade da arquitetura com relação à ativação do módulo de comunicação solicitado de acordo com a aplicação e a multiplexação de uma *interface* de comunicação apresentando dessa forma uma solução prática para um problema pouco explorado em RSSF que é a coleta remota de dados.

A Figura 4.12 apresenta o cenário montado para realização de cada estudo de caso de acordo com as especificações apresentadas.

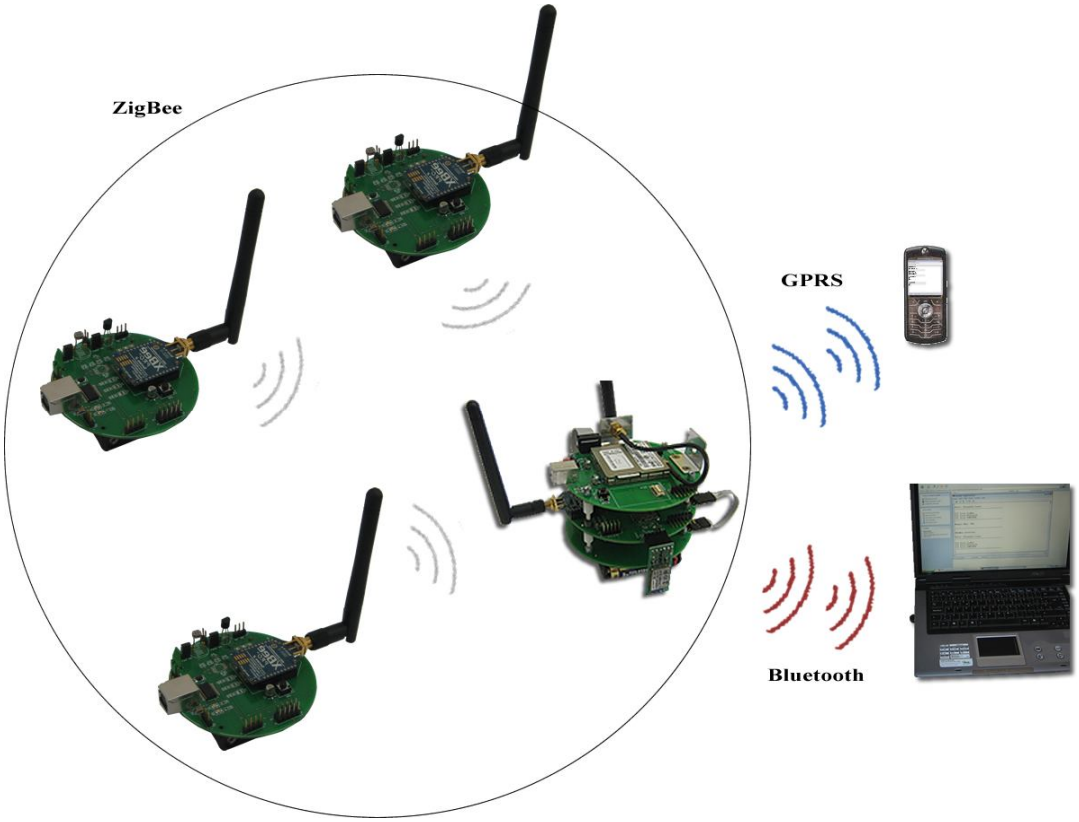


Figura 4.12 Aplicação típica para validação da arquitetura *XArmBee*.

4.4.1 Estudo de Caso 1: Coleta Presencial de Dados

A coleta presencial de dados consiste no recebimento dos dados monitorados através dos sensores existentes nos *XBeeNodes* e armazenados em um cartão de memória *SD* de *2 GB* localizado na unidade Processamento e Armazenamento de Dados da arquitetura. Os dados recebidos através da rede *ZigBee* formada entre *XBeeNodes* e o *Gateway XArmBee* configurados no modo *API* são posteriormente coletados através de um *PC* com um dongle *Bluetooth* a uma distância média de *10 m*. O processo foi realizado baseando-se nos testes realizados nas seções anteriores deste capítulo para validação da comunicação *Bluetooth*.

Os resultados obtidos foram visualmente percebidos através do recebimento dos pacotes de dados. A variação da distância entre o coletor e o *gateway* sofreu pequenas perdas, chegando a uma distância máxima de 20 m em ambiente fechado, ou seja, com paredes divisórias e obstáculos.

O objetivo principal foi atingido, com a coleta de 100% dos pacotes transmitidos. O intuito também foi validar as rotinas de ativação, multiplexação da arquitetura e o comportamento em uma situação prática. O código da aplicação é apresentado no Apêndice B.

4.4.2 Estudo de Caso 2: Coleta Remota de Dados

A coleta remota de dados consiste no recebimento dos dados monitorados no estudo de caso 1, ou seja, o mesmo cenário aplicado para realização do estudo. O diferencial é que o pacote de dados foi enviado a um coletor remoto, utilizando-se uma rede celular, através de uma mensagem *SMS*.

O validação foi baseada no algoritmo apresentado no teste do módulo *GSM/GPRS* e o resultado obtido foi satisfatório, visto que a mensagem foi recebida com sucesso. O código da aplicação é apresentado no Apêndice B.

5 Considerações Finais

Este capítulo apresenta as considerações finais acerca do que foi desenvolvido no trabalho desta dissertação tanto relativo ao trabalho em si (arquitetura desenvolvida) como ao processo que envolveu o estudo sobre o tema, a coleta de referências, estado da arte, testes e avaliações.

5.1 Conclusões

Este trabalho apresentou uma arquitetura de *hardware* modular multi-rádio que proporciona uma flexibilidade no projeto e desenvolvimento de aplicações para RSSFs, como por exemplo, a coleta presencial e remota entre redes geograficamente distribuídas verificando a convergência de dados entre redes de diferentes tecnologias de comunicação sem fio, e com uma relação de consumo de energia *versus* aplicações específicas aceitáveis tornando-se uma solução completa em um *hardware* específico.

A principal inovação consistiu em introduzir no projeto de *hardware* a capacidade de seleção entre a tecnologia de comunicação sem fio, ou seja, três rádios-transceptores foram utilizados, sendo que o *ZigBee* estava sempre ativo e os demais poderiam ser acionados dependendo da necessidade, tudo realizado na própria rede, sem a intervenção de um agente externo. Sendo que a arquitetura possibilita a expansão e adaptação de novos rádios-transceptores desde que possuam em comum a *interface* de comunicação *UART (RS-232)*.

Os estudos realizados obtiveram resultados que validam e comprovam a funcionalidade da arquitetura proposta. Foram realizadas duas avaliações, sendo uma quantitativa, verificando o consumo de energia através da medição da corrente consumida quando cada módulo de comunicação é ativado, outra qualitativa, que comparava as características de processamento da arquitetura com outras existentes.

Como resultado conclusivo verifica-se que a multiplexação de uma *interface* quer seja de controle ou de comunicação é essencial para a ampliação da capacidade de uma arquitetura de *hardware* no sentido da multi-funcionalidade.

5.2 Trabalhos Futuros

Como trabalhos futuros, sugere-se avaliar a adaptação de uma nova família de microcontroladores com mais *interfaces* seriais e um *RTOS* (*Real-Time Operating System*) para a arquitetura com *APIs* desenvolvidas visando o baixo consumo de energia.

Um estudo detalhado do desempenho (consumo de energia) da arquitetura em um cenário específico executando algoritmos para aplicações diversas.

A predição da vida útil da bateria utilizando ferramentas de *software* como: *Toolbox MATSNL,* em situações diversas, onde cada módulo seria ativado e permaneceria em atividade por um tempo definido para verificação da continuidade da aplicação.

A arquitetura já possui suporte para adaptação de novos módulos que tenham como *interface* de comunicação o padrão *UART (RS-232)* com isso sugere-se a tentativa de adaptação de módulos *Wi-Fi*, *GPS* e *HSDPA* para realização de experimentos e possíveis expansões da arquitetura.

5.3 Publicações

MARTINIANO, A. L.; FIGUEIREDO, C. M. S.; CHAVES FILHO, J. E. **XArmBee: Uma Arquitetura de Hardware Modular Multi-rádio para Gateways de Redes de Sensores Sem Fio.** *In: II Simpósio de Computação Ubíqua e Pervasiva – SBCUP 2010*, 20 – 23 Jul, Belo Horizonte – MG, Brasil.

Referências Bibliográficas

AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CYIRCI, E. **Wireless Sensor Networks: A Survey**. *Computer Networks, 2002*, 38 (4): 393-422.

ARAMPATZIS, T.; LYGEROS, J.; MANESIS, S. **A Survey of Applications of Wireless Sensors and Wireless Sensor Networks**. *In: Proceedings of the 13th IEEE Mediterranean Conference on Control and Automation (MED'05),* June 27-29, 2005, Limassol, Cyprus, pp.719-724.

BARBONI, Leonardo.; VALLE, Maurizio. **Experiemental Analysis of Wireless Sensor Nodes Current Consumption**. *In: Proceedings of the 2nd International Conference on Sensor Technologies and Applications (SENSORCOMM 2008),* August 25-31, 2008, Cap Esterel, France, pp. 401-406.

BARBOSA, T. M. G. A. **Uma Arquitetura de Redes de Sensores do Corpo Humano**. *Tese de Doutorado em Engenharia Elétrica*, *Departamento de Engenharia Elétrica, Universidade de Brasília*, Fevereiro 14, 2008, Brasília, DF, 188p.

BEUTEL, J. **Metrics for Sensor Network Platforms**. *In: Proceedings of ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, June 19, 2006, New York, pp. 26-30.

BLUETOOTH. **Bluetooth Technology**. Disponível em: http://www.bluetooth.com/English/Pages/default.aspx. Acesso em: Jan 2010.

BTNode: **A Distributed Environment for Prototyping Ad Hoc Networks**. Disponível em: http://www.btnode.ethz.ch/. Acesso em: Jan 2009.

CC2430 **System-on-Chip Solution for 2.4 GHz IEEE 802.15.4 / ZigBee™**. Disponível em: http://focus.ti.com/docs/prod/folders/print/cc2430.html. Acesso em: Fev 2009.

COULOMB **Counter/ Battery Gas Gauge**. Disponível em: http://cds.linear.com/docs/Datasheet/4150fb.pdf. 2005. Acesso em: Jun 2009.

FLASH UTILITY. **Flash Utility Software**. Disponível em: http://www.keil.com/flash/utilities.asp. 2004. Acesso em: Jan 2010.

HOLLAR, S. **COTS Dust.** *Master´s Thesis. Electrical Engineering and Computer Science Department.* UC Berkeley, December 2000.

INA138. **High-Side Measurement Current Shunt Monitor**. Disponível em http://focus.ti.com/lit/ds/symlink/ina168.pdf. Datasheet. 1999. Acesso em: Jan 2010.

JIN, Zing.; SCHURGERS, C.; GUPTA, R. K. **A gateway node with duty-cycled radio and processing subsystems for wireless sensor networks**. *In: ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 14, Issue 1, Article 5, January, 2009.*

KARL, Holger.; WILLIG, Andreas. **Protocols and architectures for wireless sensor networks**. *John Wiley & Sons*, Ltd. 2005.

KOHVAKKA, M.; ARPINEN, T.; HÄNNIKÄINEN, M.; HÄMÄLÄINEN, T. D. **High Performance Multi-radio WSN Platform**. *In: Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks (REALMAN´06)*, May 26, 2006, Florence, Italy, pp. 95-97.

LEE, Jin-Shyan.; HUANG, Yang-Chih. **ITRI ZBnode: A ZigBee/IEEE 802.15.4 Platform for Wireless Sensor Networks**. *In: Proceedings of the Systems, Man, and Cybernetics (SMC´06),*October 8-11, 2006, Taipei, China, pp. 1462-1467.

MARTINIANO JÚNIOR, E. S. **Sistema de Automação Residencial Utilizando Tecnologia ZigBee**. *Monografia (Graduação) em Engenharia da Computação, Coordenação de Engenharia da Computação, Universidade Federal do Amazonas,* Julho 16 , Manaus, AM. 2009, 87p.

MATSNL. **A MATLAB Wireless Sensor Node Platform Lifetime Prediction & Simulation Package**. Disponível em: http://enaweb.eng.yale.edu/drupal/MATSNL. Acesso em: Mar 2009.

NB100. **Stargate NetBridge**. Disponível em: http://www.xbow.com/Products/productdetails .aspx?sid=275. *Crossbow, Inc*.2007 Acesso em: Jan 2010.

NXP **LPC2141/42/44/46/48**. Disponível em: http://www.nxp.com/pip/LPC2141_42_44_46_ 48_4.html. *Datasheet*. 2005. Acesso em: Fev 2010.

ORCAD. **Cadence Orcad Solutions**. Disponível em: http://www.cadence.com/products/orca d/pages/default.aspx. 2008. Acesso em: Jan 2010.

PIDD, M. **Computer Simulation in Management Science**. 5th John Wiley & Sons, Ltd New York, 2005.

PING, Song.; CHANG, Chen.; KEJIE, Li.; LI, Sui. **The Design and Realization of Embedded Gateway Based on WSN**. *In: Proceedings of the International Conference in Computer Science and Software Engineering*, December 12-14, 2008, Wuhan, Hubei, pp. 32-36.

SCHWARZ, L. **Proposta de um Sistema Telemétrico para Aquisição de Sinais Fisiológicos**. *Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal de Santa Catarina*, Agosto 2007, Florianópolis, SC, 117 p.

SILVA, Fabrício; BRAGA, Thais R; RUIZ, Linnyer B; NOGUEIRA, José M. S. **Tecnologia de Nós Sensores Sem Fio**. *Revista Controle e Instrumentação*. 2004.

SPARKFUN. **BlueSmirf**. Disponível em: http://www.sparkfun.com/commerce/product_info. php?products_id=582. Acesso em: Jan 2010.

STARGATE. **Gateway**. Disponível em: http://www.xbow.com/Products/productdetails.aspx ?sid=229. **Crossbow, Inc**. 2004. Acesso em: Jan 2010.

TELIT. **GM862 product description**. Disponível em:<http://www.telit.com/en/products/gsm -gprs.php?p=7&p_ac=show>.2009. Acesso em: 26 set. 2009.

UBER BOARD. **Development Kit.** Disponível em: http://www.sparkfun.com/commerce/pro duct_info.php?products_id=556. Sparkfun, Inc. 2009. Acesso em: Jan 2010.

WASPMOTE. **Libelium, Inc**. 2010. Disponível em: www.libelium.com/waspmote. Datasheet. Acesso em: Jan 2010.

WEDDELL, A. S.; GRABHAM, N. J.; HARRIS, N. R. and WHITE, N. M. **Modular Plug-and-Play Power Resources for Energy-Aware Wireless Sensor Nodes**. *In: Proceedings of Sensor, Mesh and Ad Hoc Communications and Networks (SECON´09)*, June 22-26, 2009, Rome, Italy, pp. 1-9.

WINARM. **WinARM Software**. Disponível em: http://gandalf.arubi.uni-kl.de/avr_projects/winarm. 2006. Acesso em: Jan 2010.

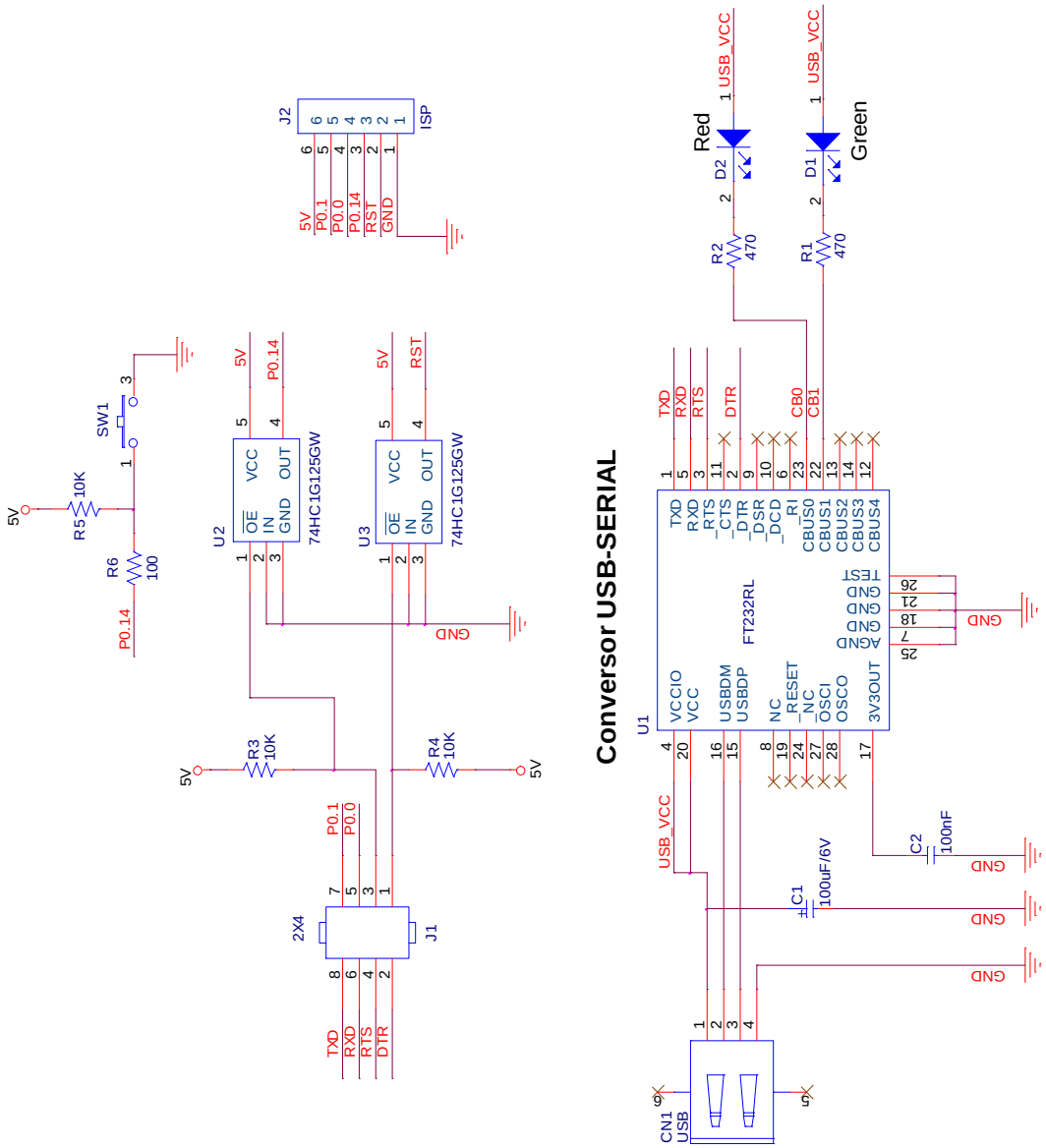
XBEE. **XBee and XBee-PRO Module**. Disponível em: http://www.digi.com/technology/wireless/products.jsp. Acesso em: Jan 2010.

X-CTU. **X-CTU Software**. Disponível em:http://www.digi.com/support/kbase/kbaseresultdetl.jsp?kb=125. 2007. Acesso em: Jan 2010.

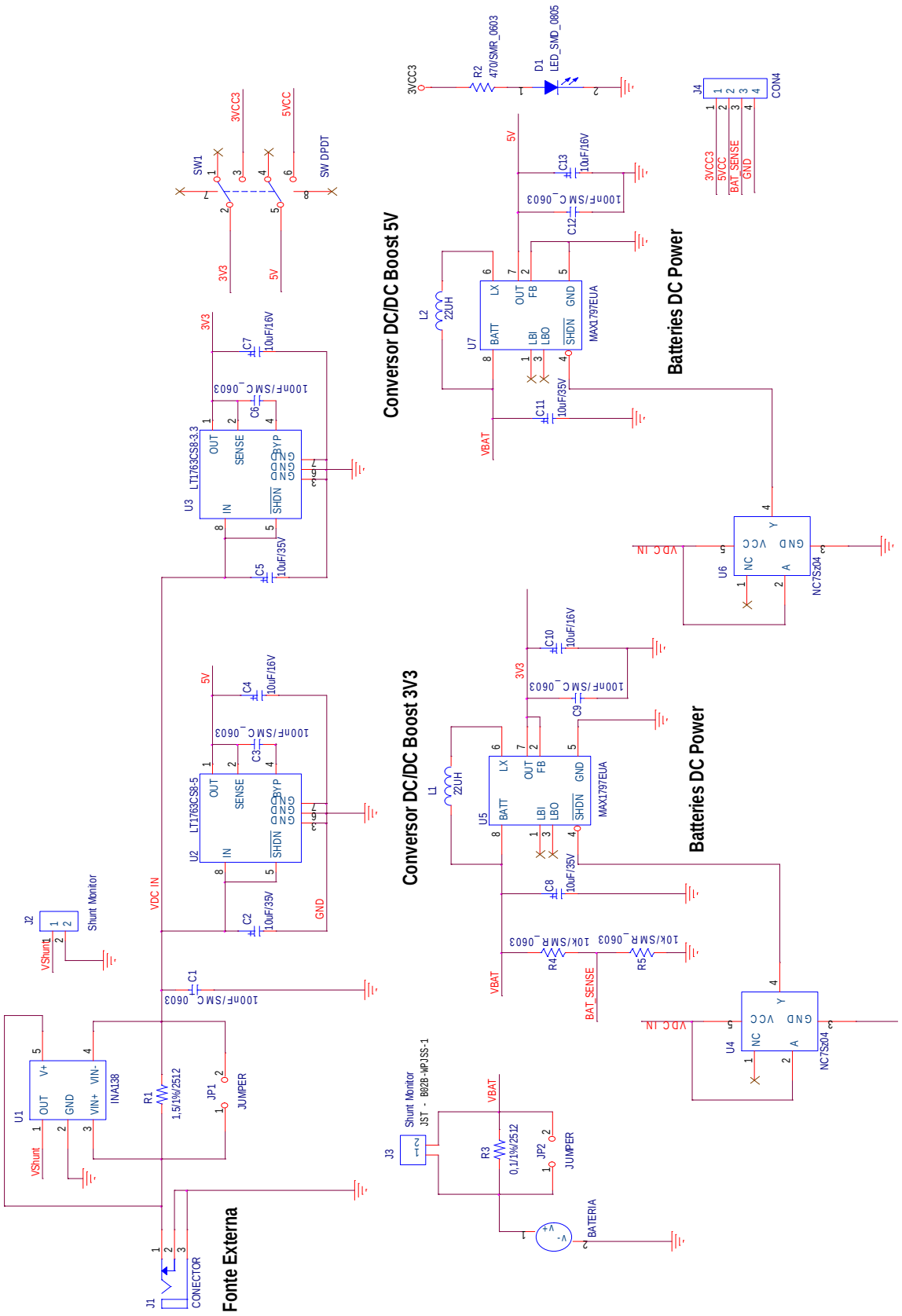
ZIGBEE. **ZigBee Alliance**.White papers. Disponível em:http://www.zigbee.org/LearnMore/WhitePapers.aspx. Acesso em: Jan 2010.

Apêndice A – Esquemáticos

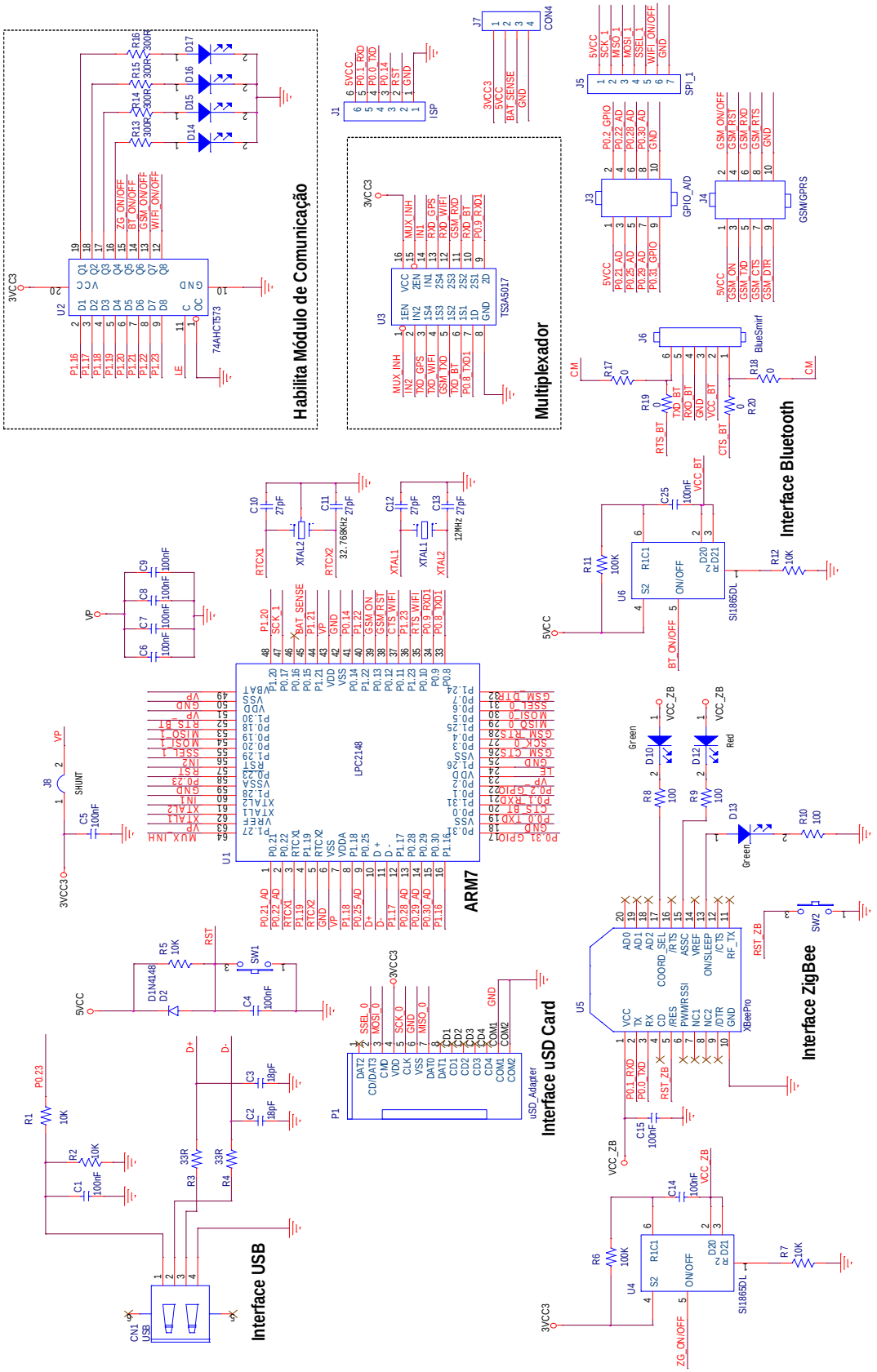
A.1 Gravador ISP



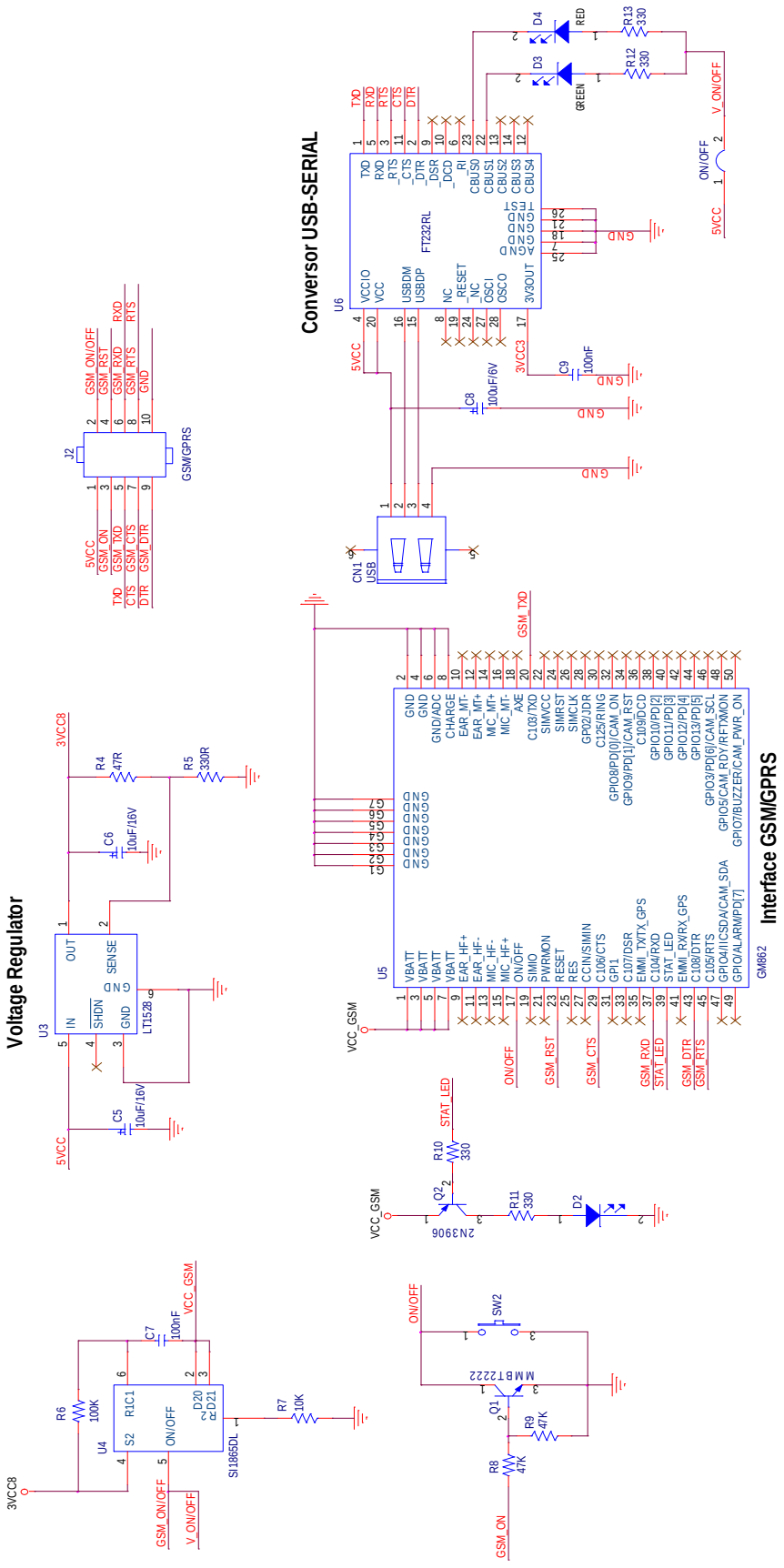
A.2 Unidade Energia



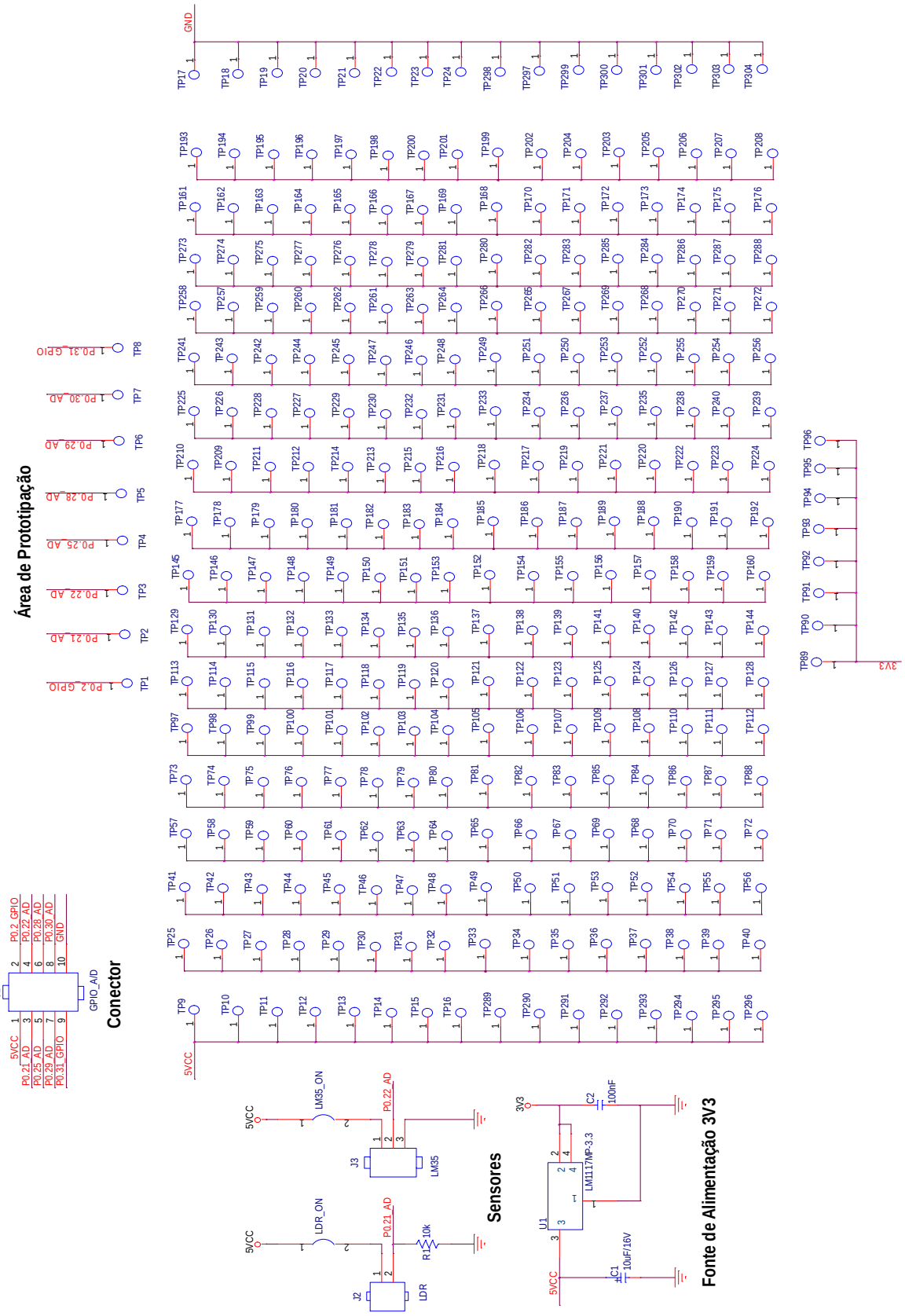
A.3 Unidade Processamento e Armazenamento de Dados



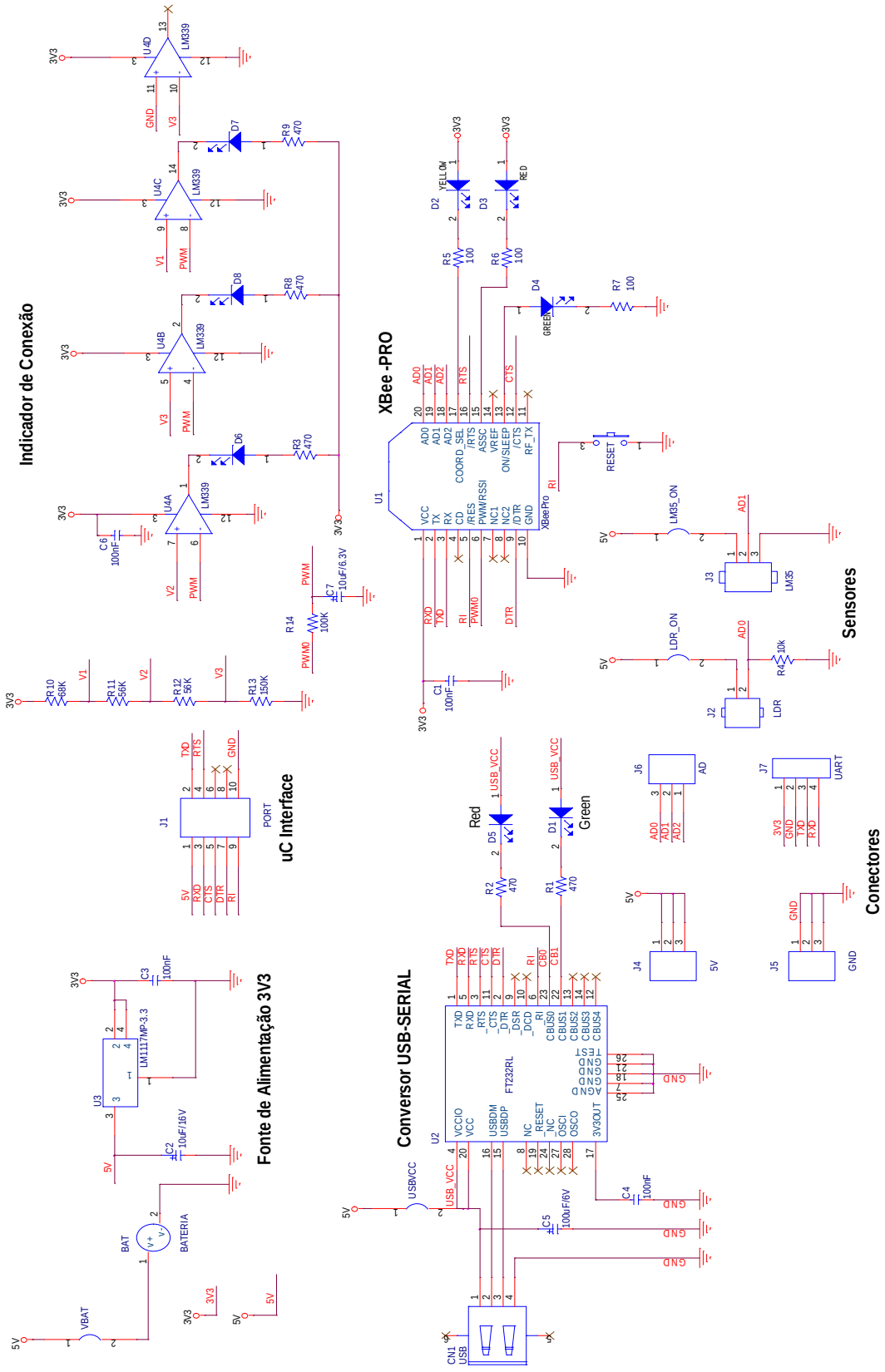
A.4 Unidade de Comunicação



A.5 Unidade Sensoriamento



A.6 *XBeeNode*


Indicador de Conexão
XBee -PRO
Sensores
uC Interface
Fonte de Alimentação 3V3
Conversor USB-SERIAL
Conectores
Red
Green
PORT
UART
BATERIA
LM35
LDR

Apêndice B – Códigos Fonte

```
//*************************************************************
// Autor: Alexandre Lopes
// XArmBee - LPC2148
//*************************************************************


/* ********************************************************
              Declaração de Funções
  ******************************************************** */

void Initialize(void);
void feed(void);

void put_char(char c);
char getc(void);

void setup_uart0(int baud);
void setup_uart1(int baud);

void fat_initialize(void);
void delay_ms(int count);
void config_menu(void);

/********************************************************
                  Bibliotecas
 ********************************************************/
#include "LPC214x.h"
#include "rprintf.h"
#include "sysTime.h"
#include "fat16.h"
#include "sd_raw.h"
#include "xbee.h"



/********************************************************
                  Variáveis Globais
 ********************************************************/
char port = 0;        //determina qual port será escrito
char channel = 0;

/********************************************************
                      Programa Principal
********************************************************/

int   main (void)
{
      Initialize();

      rprintf_devopen(put_char); /* init rrprintf */

      setup_uart0(9600);

      rprintf("Iniciando o Gateway XArmBee...\r\n",0);

      setup_uart1(9600);

      config_menu();

      while(1);
```

```
}


/*********************************************************
                      Inicialização
*********************************************************/

#define PLOCK 0x400

void Initialize(void)
{
        // Setting Multiplier and Divider values
      PLLCFG=0x23;
      feed();

      // Enabling the PLL */
      PLLCON=0x1;
      feed();

      // Wait for the PLL to lock to set frequency
      while(!(PLLSTAT & PLOCK)) ;

      // Connect the PLL as the clock source
      PLLCON=0x3;
      feed();

      // Enabling MAM and setting number of clocks used for Flash memory fetch
      MAMCR=0x2;
      MAMTIM=0x4;

      // Setting peripheral Clock (pclk) to System Clock (cclk)
      VPBDIV=0x01;

      PINSEL0 = 0x00050005;    // enable uart0, enable uart1

}


void feed(void)
{
  PLLFEED=0xAA;
  PLLFEED=0x55;
}


void put_char(char c)
{
    if (port == 0)
    {
        while((U0LSR & 0x20) == 0);
        U0THR = c;
    }
    else if (port == 1)
    {
        while((U1LSR & 0x20) == 0);
        U1THR = c;
    }
}

//specific to UART0
char getc(void)
```

```
{
    while((U0LSR & 0x01) == 0);
    return (U0RBR);
}
/*********************************************************
                    Configuração da UART 0
*********************************************************/

void setup_uart0(int baud)
{

      //set up uart0
      U0LCR = 0x83;               // 8 bits, no Parity, 1 Stop bit, DLAB = 1


      if (baud == 1200)
      {
            U0DLM = 0x09;
            U0DLL = 0xC0;         // 1200 Baud Rate @ 48MHz VPB Clock
      }

      if (baud == 2400)
      {
            U0DLM = 0x04;
            U0DLL = 0xE0;         // 2400 Baud Rate @ 48MHz VPB Clock
      }

      if (baud == 4800)
      {
            U0DLM = 0x02;
            U0DLL = 0x70;         // 4800 Baud Rate @ 48MHz VPB Clock
      }

      else if (baud == 9600)
      {
            U0DLM = 0x01;
            U0DLL = 0x38;         // 9600 Baud Rate @ 48MHz VPB Clock
      }

      else if (baud == 19200)
      {
            U0DLM = 0x00;
            U0DLL = 0x9C;         // 19200 Baud Rate @ 48MHz VPB Clock
      }

      else if (baud == 38400)
      {
            U0DLM = 0x00;
            U0DLL = 0x4E;         // 38400 Baud Rate @ 48MHz VPB Clock
      }

        else if (baud == 57600)
      {
            U0DL  = 0x00;
            U0DLL = 0x34;         // 57600 Baud Rate @ 48MHz VPB Clock

      }

        else if (baud == 115200)
      {
            U0DLM = 0x00;
            U0DLL = 0x1A;         // 115200 Baud Rate @ 48MHz VPB Clock
```

```
	}

	U0FCR = 0x01;
	U0LCR = 0x03;		// DLAB = 0
}

/*****************************************************
			Configuração da UART 1
*****************************************************/


void setup_uart1(int baud)
{
	//set up uart1
	U1LCR	= 0x83;		// 8 bits, no Parity, 1 Stop bit, DLAB = 1

	if (baud == 1200)
	{
		U1DLM = 0x09;
		U1DLL = 0xC0;		// 1200 Baud Rate @ 48MHz VPB Clock
	}

	if (baud == 2400)
	{
		U1DLM = 0x04;
		U1DLL = 0xE0;		// 2400 Baud Rate @ 48MHz VPB Clock
	}

	if (baud == 4800)
	{
		U1DLM = 0x02;
		U1DLL = 0x70;		// 4800 Baud Rate @ 48MHz VPB Clock
	}

	else if (baud == 9600)
	{
		U1DLM = 0x01;
		U1DLL = 0x38;		// 9600 Baud Rate @ 48MHz VPB Clock
	}

	else if (baud == 19200)
	{
		U1DLM = 0x00;
		U1DLL = 0x9C;		// 19200 Baud Rate @ 48MHz VPB Clock
	}

	else if (baud == 38400)
	{
		U1DLM = 0x00;
		U1DLL = 0x4E;		// 38400 Baud Rate @ 48MHz VPB Clock
	}

	  else if (baud == 57600)
	{
		U1DLM = 0x00;
		U1DLL = 0x34;		// 57600 Baud Rate @ 48MHz VPB Clock
	}

	  else if (baud == 115200)
	{
		U1DLM = 0x00;
		U1DLL = 0x1A;		// 115200 Baud Rate @ 48MHz VPB Clock
	}
```

```
      U1FCR = 0x01;

      U1LCR = 0x03;           // DLAB = 0
}

/*********************************************************
     Função de Sinalização – Ativação - Multiplexação
*********************************************************/


void liga16_17_20_21_26(void) // ZigBee e Bluetooth
{

  PINSEL1 = 0x00000000; //Configura os pinos da porta 1 como GPIO
  IO1DIR  = 0x34FF0000; //Configura os Pinos P1.26 e de P1.16 a P1.23 como Saída
  IO1SET  = 0x04330000; // Seta os pinos P1.26, P1.16, P.17, P1.20 e P1.21 alto
  IO1CLR  = 0x0BCC0000; // Seta os demais em baixo

}

void liga16_18_20_22_26_29(void) // ZigBee e GSM/GPRS
{

  PINSEL1 = 0x00000000;
  IO1DIR  = 0x34FF0000;
  IO1SET  = 0x24550000;
  IO1CLR  = 0xDBAA0000;
}

/*********************************************************
                    Funções do SD Card
*********************************************************/

void fat_initialize(void)
{
      if(!sd_raw_init())
      {
            rprintf("SD Init Error\n\r");
            while(1);
      }

      if(openroot())
      {
            rprintf("SD OpenRoot Error\n\r");
      }
}

void createLogFile(void){

      static int file_number;

      rprintf(file_name, "sensor_read.txt", file_number);

    while(root_file_exists(file_name))
    {
        file_number++;

        if(file_number == 250)
        {
            //rprintf("\n Arquivo completo!\n");
        }
        rprintf(file_name, "sensor_read.txt", file_number);}
```

```
/*********************************************************
                        Função Delay
*********************************************************/

void delay_ms(int count)
{
      int i;
      count *= 10000;
      for(i = 0; i < count; i++)
            asm volatile ("nop");
   }

/*********************************************************
                        Função BlueSmirf
*********************************************************/
void sendBluetooth(void)
{
      rprintf_devopen(put_char);    //Init rprintf on UART1 (Bluetooth Channel)
      setup_uart1(9600);            //Set bluetooth Baud Rate
      BluetoothOn();                //Turn on the Bluetooth device
      delay_ms(200);
      configBluetooth();            //Send configuration strings
   }
void configBluetooth(void)
   {
            //Enter Command Mode
            delay_ms(100);
            rprintf("$$$");
            delay_ms(100);
            //Disable the Config Timer
            rprintf("ST,0\r");
            delay_ms(50);
            //Activate Fast Data Mode
            rprintf(SN, "sensor_read.txt", \r);
            delay_ms(50);
            //Activate Fast Data Mode
```

```
            rprintf("F,1\r");

            delay_ms(50);

   }


/*********************************************************
                      Função GSM/GPRS
*********************************************************/
void sendSMS(void)

   {

      rprintf_devopen(put_char);    //Init rprintf on UART1 (GM862 Channel)

      setup_uart1(9600);            //Set GM862 Baud Rate

      //Enter AT Commands to Send SMS

      rprintf("AT+CMGF = 1\r\n"); //Set SMS to text mode

      delay_ms(300);

      rprintf("AT+CMGS =");//Set a valid phone number

      delay_ms(300);

      rprintf("+9281329584\r\n");

      delay_ms(300);

      SMS_upload("sensor_read.txt");

      delay_ms(300);

   }


void config_menu(void)
{
      char a, q;


      while (1)
      {
            rprintf("=======================================\r\n");
            rprintf("\r\n\XArmBee Interface\n");
            rprintf("\r\n\Autor: Alexandre Lopes\r\n");
            rprintf("=======================================\r\n\n");
            rprintf("(1) Teste ZigBee e Bluetooth\r\n");
            rprintf("(2) Teste ZigBee e GSM/GPRS\r\n");
            rprintf("(3) Teste SD Card");
            rprintf("=======================================\r\n");

            a = getc();

            if (a == '1')
            {
                  rprintf("\r\n\Modulos XBee  PRO e BlueSmirf\n\n");
                  liga16_17_20_21_26();
                  xbee_init();
                  delay_ms(300);
                  xbee_message();
                  sendBluetooth();
```

```
            }

            else if (a == '2')
            {
                  rprintf("\r\n\Modulos XBee PRO e Telit GM862\n\n");
                  liga16_18_20_22_26_29();
                  xbee_init();
                  delay_ms(300);
                  xbee_message();
                  sendSMS();
            }
            else if (a == '3')
            {
                  rprintf("\r\n\r\n",0);
                  m = sd_raw_init();
                  createLogFile();
                  if (m == 1) rprintf("Success\r\n",0);
                  else if (m == 0) rprintf("Failed\r\n",0);
            }


      }

}
```