PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS

PROGRAMADE PÓS-GRADUAÇÃO STRICTO SENSU - MESTRADO EM SISTEMA DE INFRAESTRUTURA URBANA

RAFAEL GONÇALVES TOZZO

DESENVOLVIMENTO E TESTE DE NÓ SENSOR SOLAR PARA RSSF

CAMPINAS

2017

RAFAEL GONÇALVES TOZZO

DESENVOLVIMENTO E TESTE DE NÓ SENSOR SOLAR PARA RSSF

Dissertação apresentada como exigência para obtenção do Título de Mestre em Infraestrutura Urbana, ao Programa de Pós-Graduação em Sistemas de Infraestrutura Urbana, do Centro de Ciências Exatas, Ambientais e de Tecnologia, da Pontifícia Universidade Católica de Campinas.

Orientador: Prof. Dr. Omar Carvalho Branquinho

PUC-CAMPINAS

2017

Ficha Catalográfica
Elaborada pelo Sistema de Bibliotecas e
Informação - SBI - PUC-Campinas

t6221.3851
T757d

Tozzo, Rafael Gonçalves.
Desenvolvimento e teste de nó sensor solar para RSSF / Rafael Gonçalves Tozzo.- Campinas: PUC-Campinas, 2017.
138 p.

Orientador: Omar Carvalho Branquinho.
Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-Graduação em Sistemas de Infraestrutura Urbana.
Inclui anexo e bibliografia.

1. Redes de sensores sem fio. 2. Sistemas de computação sem fio 3. Ambientes externos. 4. Sistemas de transmissão de dados. 5.Tecnologia da informação. 6. Detectores. I. Branquinho, Omar Carvalho. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Sistemas de Infraestrutura Urbana. III. Título.

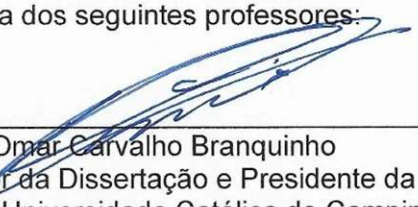
22.ed. CDD – t621.3851

RAFAEL GONÇALVES TOZZO

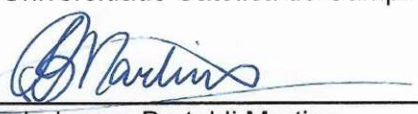
DESENVOLVIMENTO E TESTE DE NÓ SENSOR SOLAR PARA RSSF

Dissertação apresentada ao Curso de Mestrado em Sistemas de Infraestrutura Urbana do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Sistemas de Infraestrutura Urbana.
Área de Concentração: Sistemas de Infraestrutura Urbana.
Orientador (a): Prof. (a) Dr. (a) Omar Carvalho Branquinho.
Co-orientação: Prof. (a) Dr. (a) Lia Toledo Moreira Mota.

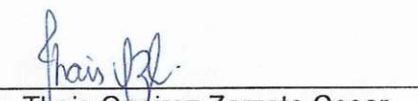
Dissertação defendida e aprovada em 29 de junho de 2017 pela Comissão Examinadora constituída dos seguintes professores:



Prof. Dr. Omar Carvalho Branquinho
Orientador da Dissertação e Presidente da Comissão Examinadora
Pontifícia Universidade Católica de Campinas



Profa. Dra. Indayara Bertoldi Martins
Pontifícia Universidade Católica de Campinas



Profa. Dra. Thais Queiroz Zorzeto Cesar
Universidade Estadual de Campinas - UNICAMP

Dedico este trabalho a Deus, que sempre me deu força e sabedoria. Em momentos em que tudo se parecia perdido, Ele me mostrou o caminho, sempre me confortando diante das dificuldades, tornando real a concretização de mais uma conquista.

AGRADECIMENTOS

Ao Prof. Dr. Omar Carvalho Branquinho,
Orientador e incentivador, guia e mestre sempre atento e aplicado na minha formação profissional e amigo sincero em todos os momentos.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão de bolsa durante o período de realização deste mestrado

“Maravilhas nunca faltaram ao mundo; o que sempre falta é a capacidade de senti-las e admirá-las”

Mário Quintana

(1906-1994)

RESUMO

TOZZO, Rafael Gonçalves. DESENVOLVIMENTO E TESTE DE NÓ SENSOR SOLAR PARA RSSF. 2017. 138p. Dissertação (Mestrado em Infraestrutura Urbana) - Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-graduação em Sistema de Infraestrutura Urbana, Campinas, 2017.

A dissertação descreve o desenvolvimento de um elemento de rede sem fio autônomo, que possibilita a pesquisadores a coleta de informações em ambientes externos. Investigando a autonomia de nós sensores, e o dimensionamento de uma bateria e de um painel fotovoltaico. Para a validação da solução, utiliza-se o nó sensor desenvolvido para a realização de coletas de grandezas em áreas externas por períodos de tempo. Aplicando estratégias de dimensionamento de um sistema de energia visando as necessidades de um nó sensor em ambiente externo. Foi desenvolvida uma estratégia de economia de energia, com *sleep mode* que permite ciclos de funcionamento maiores ao NSS. Os resultados é a constatação de que o NSS funcionou no período, através do nível de tensão da bateria, que está relacionado com a energia da mesma. Podendo afirmar-se que o NSS funcionou e atente ao objetivo de monitorar grandezas em ambientes externos.

Termos de indexação: Rede via rádio, RSSF, PV, Nó Sensor Autônomo

ABSTRACT

TOZZO, Rafael Gonçalves. DEVELOPMENT AND TEST OF WSN SOLAR SENSOR NODE. 2017. 138p. Qualifying thesis (Master in Urban Infrastructure System) - Pontifical Catholic University of Campinas, Centre for Mathematical Sciences, Environmental and Technology, Graduate in Electrical Engineering, Campinas, 2017.

The dissertation describes the development of an autonomous wireless network element, which enables researchers to gather information in outdoor environments. Investigating the autonomy of sensor nodes, and the dimensioning of a battery and a photovoltaic panel. For the validation of the solution, is used the sensor node developed in this work to collect data in external areas for periods of time. Applying strategies for dimensioning an energy system to aiming the needs of a sensor node in an external environment. An energy-saving strategy has been developed with sleep mode that allows higher work cycles to the NSS. The results are the finding that the NSS worked in the period, through the voltage level of the battery, which is related to the energy of the same. It can be affirmed that the NSS functioned and the objective of monitoring quantities in outdoor environments

Indexing terms: Network radio, WNS, PV, Standalone Sensor Node.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RSSF.....Rede de Sensores Sem Fio

UART.....*Universal Asynchronous Receiver/Transmitter*

IDE……*Integrated Development Environment*

IEEE….....Instituto de Engenheiros Eletricistas e Eletrônicos

PV.....*PhotoVoltaic* (Painel Fotovoltaico)

MPPT.....*Maximum Power Point Tracking*

RS.....Rede de Sensores

EH……*Energy Harvest*

GUI……..*Graphical User Interface*

PC……*Personal Computer*

FTDI……*Future Technology Devices International*

NSS……Nó Sensor Solar

MAC……*Media Access Control* (Controle de Acesso ao Meio)

NS.....Nó Sensor

RX.....Receptor

TX.....Transmissor

SUMÁRIO

1 INTRODUÇÃO

Com a crescente demanda por ações de sustentabilidade (JACOBI, 2003), buscando conciliar conforto com preservação ambiental, novos processos, que prometem aumentar a eficiência de atividades exercidas no meio ambiente, necessitam de um histórico das grandezas que se referem ao meio. São informações que possibilitam representar o ambiente, coletadas através de sensores, usadas para realizar simulações computacionais (MEDEIROS; MEDEIROS, 2002) ou avaliar o meio ambiente.

Sensores são transdutores de grandezas, que permitem que fenômenos físicos sejam convertidos em sinais elétricos. Atualmente, existe uma enorme gama de transdutores, com os quais é possível converter até gestos humanos (MORAES; TARRACHELLI, 2010) em sinais elétricos. Para a monitoração de diversos pontos, como o mapeamento térmico de um cômodo, de forma a coletar em diversos pontos a mesma grandeza, faz-se o uso de Redes de Sensores Sem Fio (RSSF). Uma RSSF tem nós que se comunicam por meio de radiofrequência, e tem como função mapear o local instalado através das grandezas coletadas.

Nó Sensor Solar (NSS) é um nó sensor de RSSF com as mesmas atribuições, porem este tem sua autonomia energética. Através de NSS oferecer uma ferramenta para ciências que necessitam de monitorações ou coleta de dados

Havendo interesse de coleta de grandezas em um ambiente externo, desprovido de uma infraestrutura fixa de energia elétrica, que possibilite a melhora no gerenciamento ou na sustentabilidade do mesmo. Com isto trazendo melhorias para o meio urbano, pois é uma ferramenta a mais para os gestores avaliarem o que realmente acontece.

Com a chegada da Internet das coisas, a RSSF é uma ferramenta essencial e a criação de um nó sensor autônomo, do ponto de vista de alimentação de energia, aumenta o potencial de coleta de dados no ambiente. O NSS se torna um subsidio para a internet das coisas, coletando as grandezas do ambiente (GUBBI et al., 2013).

Estratégias de minimização do consumo do NSS podem ser aplicadas para o aumento da vida da rede. Uma destas estratégia é a Controle de Acesso ao Meio (MAC), que possibilita que o NSS permaneça mais tempo em modo que consuma menos energia (RUIZ et al., 2004). Para uma maximização da vida útil da rede, se faz necessário o

uso de estratégias que são implementadas conforme a aplicação ou são norteadas por aplicações similares já existentes.

1.1 OBJETIVO

O objetivo é desenvolver um NSS.

Para desenvolver um NSS é necessário saber, *Energy Harvest* (EH), armazenamento de energia, consumo de carga, empacotamento hardware/mecânico e protocolo de comunicação.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

O Nó Sensor (NS) é formado por um conjunto de elementos abordado no Capítulo 2, que quando implementado seja o mais simples possível (RIBEIRO, 2010). A simplicidade desejada tem como objetivo fazer com que o custo e o tamanho do NS sejam os menores possíveis. No Capítulo 2 apresenta-se uma fundamentação teórica, com as informações necessárias para o entendimento do trabalho, abordando conceitos de EH, Painel fotovoltaico (PV), baterias, RSSF, cálculo para o dimensionamento do sistema de energia do NSS e um resumo de antenas. No Capítulo 3 está a proposta desta dissertação, se refere ao NSS. O Capítulo 4 apresenta os materiais utilizados no andamento das atividades práticas, como foram realizados os testes práticos e os métodos empregado. O Capítulo 5 apresenta os resultados obtidos pelos experimentos práticos realizados, juntamente com a análise dos resultados obtidos, observando o que representa os dados coletados para o desenvolvimento do NSS. No Capítulo 6 está a conclusão

2 FUNDAMENTAÇÂO TEÓRICA

Este capítulo apresenta os conceitos teóricos referentes aos que deram suporte ao desenvolvimento de trabalho. Conhecimentos sobre a coleta de energia, tecnologias envolvidas no desenvolvimento de uma RSSF e exemplos desse tipo de rede encontrado na literatura.

2.1 ENERGY HARVESTING

Energy Harvest (EH) é o processo de aproveitamento de energia disponível no ambiente. Pode ser obtida por calor, luz, som, vibração ou movimento (MACCHIARELLI, [s.d.]) e quando não utilizada, é perdia.

O uso de EH para aplicações em redes sem fio vem ganhando uma atenção considerável, permitindo que as técnicas se tornem mais robustas ao longo do tempo. O modelo, normalmente usados, traz a sequência: a energia do ambiente, *harvest*, armazenamento, sensor de energia, controlados e a carga (ABBAS et al., 2014).

- **Energia do Ambiente:** é a energia que está no meio ambiente, como ruídos de estradas, calor do deserto, vento, energia solar entre outros.
- ***Harvesting*:** é um termo em inglês designado para colheita, no infinitivo. No que se refere ao trabalho é o ato de, literalmente, colher a energia disponível do meio com alguma ferramenta de coleta, como PV, turbinas entre outras.
- **Armazenamento:** continuando a analogia de colheita, usam-se armazéns que guardam a energia coletada, como, baterias, super capacitores ou outros.
- **Sensor de energia:** útil para quantificar a energia gerada e armazenada, fornecendo informações para tomada de decisão.
- **Controlador:** responsável por aplicar os regimes de carga e descarga dos mecanismos de armazenamento, praticamente um gerente de energia.
- **Carga:** o ponto de interesse do sistema no armazém.

Segundo Raghunathan et. al. (2005) as tecnologias de EH proporcionam uma solução para a autonomia energética cada vez mais robusta para vários dispositivos, como roupas inteligentes, acessórios de suporte à vida, RSSF, entre outros. Para se ter uma proporção dessas energias (RF MONOLITHICS, 2010), a Tabela 1 apresenta o potencial de energia obtido pelas principais fontes de EH. A energia disponível no

ambiente de diversas formas, sendo que mais atrativa destas fontes é a célula solar, com eficiência de colheita superior às demais.

Tabela 1 – Capacidade de colheita de energia das principais fontes de EH

HARVESTING TECHNOLOGY	POWER DENSITY
Célula solar	15 mW/cm3
Piezoelétrico	330 μW/cm3
Vibração	116 μW/cm3
Termoelétrico	40 μW/cm3
Ruído acústico	960 nW/cm3

Fonte: Adaptado de RAGHUNATHAN et al., (2005)

2.2 GERAÇÃO FOTOVOLTAICA

A geração fotovoltaica consiste basicamente em um fenômeno que consiga converter energia luminosa solar em energia elétrica, através de placa solar.

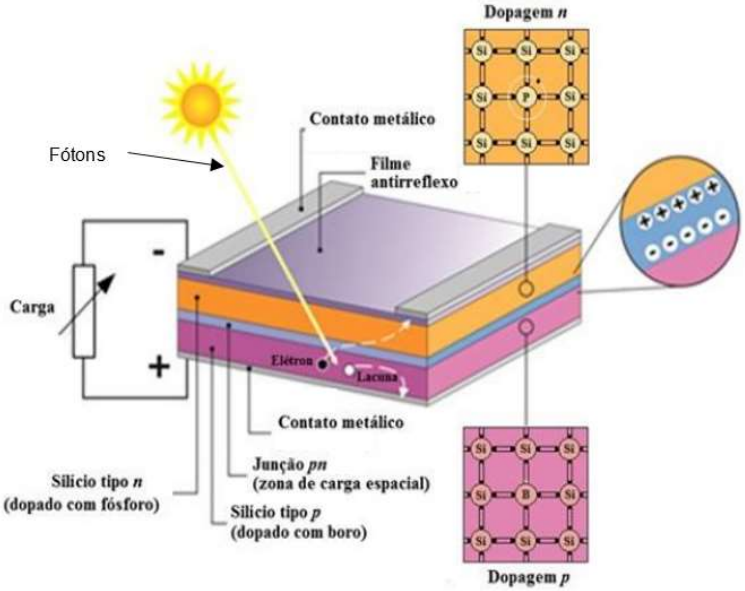
Segundo Einstein (1921, Efeito fotoelétrico)

> "Certos corpos emitem elétrons quando atingidos pela luz, porque a luz não é uma onda continua, mas composta por pequenas partículas de energia.
>
> Fótons ou quantum de luz"

A luz é composta por pacotes de energia, quantum, que recebem a denominação de fótons. Esses pacotes ao incidirem em uma célula fotovoltaica fazem com que os elétrons circulem de uma camada para outra, fluindo uma corrente elétrica (PINTO et al., 2015, p.16).

As células fotovoltaicas são constituídas de camadas de silício dopado. As células possuem uma camada N, que contém mais elétrons na camada de valência, e uma camada P, que possui menos elétrons na camada de valência. Entre as camadas há um meio condutor, no qual os elétrons transitam quando há incidência de luz, polarizando a célula (Figura 1) (MORALES, 2010).

Figura 1 – Principio físico da célula fotovoltaica



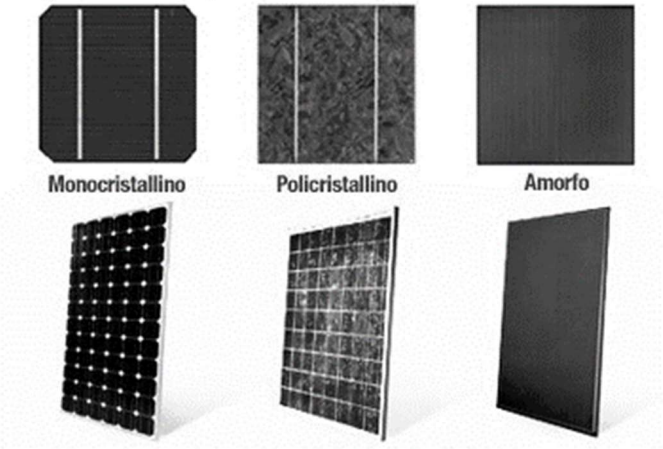
Fonte: Adaptada de PINHO e GALDINO (2014)

A célula fotovoltaica em sua essência tem as duas camadas principais de silício, com dopagens diferentes. As dopagens são responsáveis por fazer a lacuna na camada de valência. Quando o fóton incide na célula, surge uma corrente quando uma carga estabelece a conexão entre as principais camadas. Caso a célula não possua carga, a energia potencial aumenta, ocasionando a polarização das camadas, assim apresenta um comportamento semelhante a uma pilha, anulando o fluxo da corrente.

Devido ao processo de fabricação, existem atualmente três principais tipos de células: amorfa, mono cristalina e poli cristalina (Figura 2). As mais utilizadas são as cristalinas, pois possuem uma melhor eficiência de conversão energética entre 15 a 25 % (NASCIMENTO, 2004)

Os objetivos atuais em pesquisas com células fotovoltaicas é fazer células flexíveis e com um melhor eficiência (MIGUEL e GOMES, 2012).

Figura 2 – Tipos de células fotovoltaicas



Fonte: MARJOYA (2016)

A placa solar é um arranjo de células fotovoltaicas, que tem a capacidade de prover um potencial energético pela incidência de energia solar.

Um conjunto de células aumenta potencial, e deve ser dimensionado de acordo com a demanda necessária.

Quando se trata de geração é preciso, antes da instalação do PV, fazer uma análise da área onde será implantado. Há muitas variáveis que compõem o sistema de geração, entre elas a carta solar da região, o histórico climático, posição da região no globo terrestre, entre outras (SALAMONI, 2004). Com as informações do terreno é possível saber para qual direção o PV deve estar apontado para ficar maior tempo possível perpendicular aos raios solares, colhendo a energia solar de forma mais efetiva. O ângulo do PV deve estar posicionado referente ao solo, para aumentar a intensidade da irradiação, provendo uma melhor eficiência. Nos Emirados Árabes tem se utilizado um conjunto de espelhos para aumentar essa intensidade (GROSSMANN, 2013).

STELLBOGEN (1993) desenvolveu um modelo de circuito elétrico para melhor compreensão da célula fotovoltaica (Figura 3), que descreve as características da célula com equivalentes em componentes eletrônicos onde:

I_g → É uma fonte de corrente dependente de fótons provedora de energia do circuito.

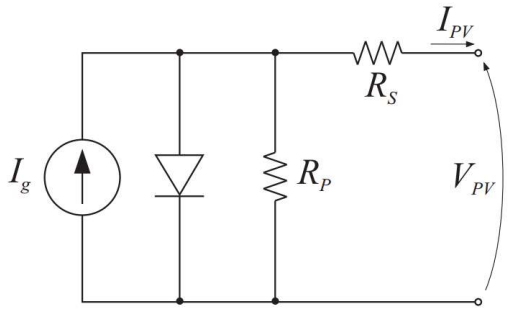
R_P → É uma resistência em paralelo, mostra que há energia potencial na célula mesmo quando o circuito estiver em aberto, quando I_{PV} =0 (Tornando a em R_P = V_{PV}).

R_S → É uma resistência em série, que representa uma resistência interna que dissipa energia.

V_{PV} e I_{PV} → São os parâmetros de corrente e tensão fornecidos para uma carga acoplada à célula.

Para a obtenção de uma célula com melhor desempenho, busca-se uma R_P tendendo a infinito e uma R_S tendendo a 0 (ELIANE et. al., 2004).

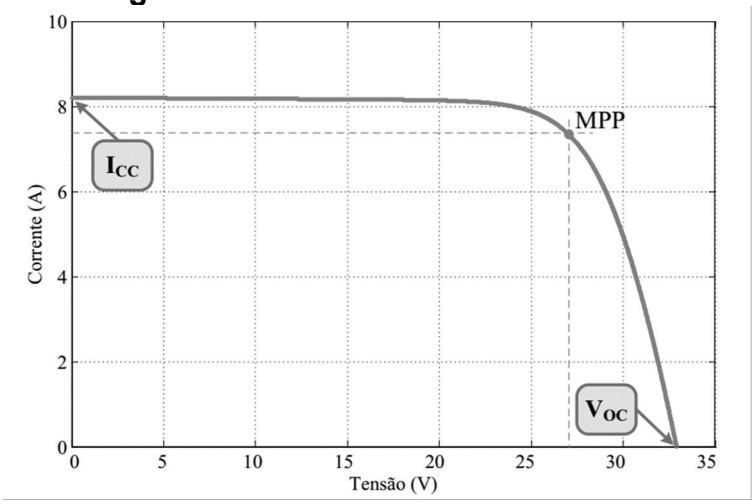
Figura 3 – Circuito equivalente de uma célula fotovoltaica



Fonte: AZEVEDO et al.(2008)

BRITO e LUIGI (2010) determinam a curva característica de um PV (Figura 4), cujo os principais destaques são a corrente de curto circuito (ICC), a tensão de circuito aberto (VOC) e o ponto de *Maximum Power Point* (MPP) que é o maior produto de ICC e VOC, resultante da energia máxima obtida pela célula.

Figura 4 – Curva característica de um PV



Fonte: BRITO e LUIGI (2010)

Onde:

I_{CC} → É a corrente de curto circuito (*Short Circuit*), quando a carga do circuito da Figura 3 é 0, proporcionando a corrente máxima

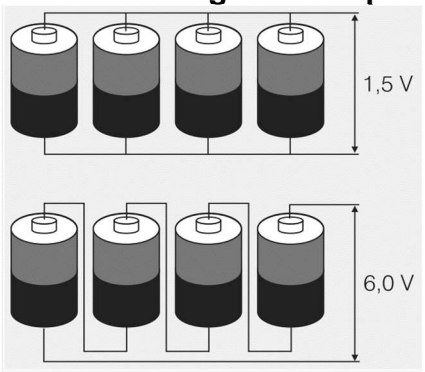
V_{OC} → É a tensão de circuito aberto (*Open Circuit*) quando a carga do circuito da Figura 3 tende a infinito, proporcionando uma corrente nula.

MPP se refere ao ponto de máxima potência, que é obtido excursionando a carga ligada ao PV, onde almeja-se o maior MPP possível.

2.3 BATERIA

As baterias é uma composição de pilhas que é composta por dois eletrodos e um eletrólito. Se ligadas em série, aumentam sua energia potencial; e, em paralelo são capazes de fornecer corrente com o mesmo potencial (Figura 5). O eletrólito usado é do tipo iônico, podendo ser de diversos materiais condutores (BOCCHI et al. 2000).

Figura 5 – Potencial energético de pilhas alcalinas



Fonte: BOCCHI et al., (2000)

O arranjo das pilhas determina a capacidade da bateria, sendo que cada pilha possui 1,5 V e 2 Ah. O primeiro arranjo Figura 4 é equivalente a uma bateria de 1,5 V e 8 Ah; adicionando mais uma pilha neste mesmo arranjo, a bateria aumentaria a capacidade de armazenamento para 10 AH, a bateria equivalente do segundo arranjo é de 6 V e 2 Ah e adicionando mais uma pilha, a energia potencial passaria para 7,5 V; para atender uma necessidade de uma bateria de 12 V e 10 AH, utilizando pilhas de 1,5 V e 2 AH, seria composta de 5 arranjos de 8 pilhas em série ligadas em paralelo.

Dentre as baterias, existem as primárias e as secundárias. Basicamente, as primárias não são recarregáveis, como exemplo as alcalinas que são compostas de eletrólitos de zinco/dióxido de manganês. As secundárias são recarregáveis, com uma capacidade mínima de 300 ciclos de carga e descarga com 80% de sua capacidade nominal, ou seja, até os 300 ciclos a capacidade da bateria terá no mínimo 80% dos AH nominal quando recarregada. Como exemplo de baterias secundárias pode ser mencionada a bateria de chumbo/ácido com eletrólitos de chumbo/óxido de chumbo (BOCCHI et al., 2000). Dentre as secundárias, as mais comuns são as: chumbo/óxido de chumbo (chumbo/ácido), cádmio/óxido de níquel (níquel/cádmio), hidreto metálico/óxido de níquel e íons lítio.

O dimensionamento da bateria para aplicações com EH solar é realizado de acordo com a capacidade mínima da bateria. Como há uma variação na captação da EH solar, considera-se o intervalo de tempo sem incidência solar disponível (Equação 2.1) (WIN et al., 2010).

$$B_C = I_M \times T_{SS} \qquad (2.1)$$

Onde:

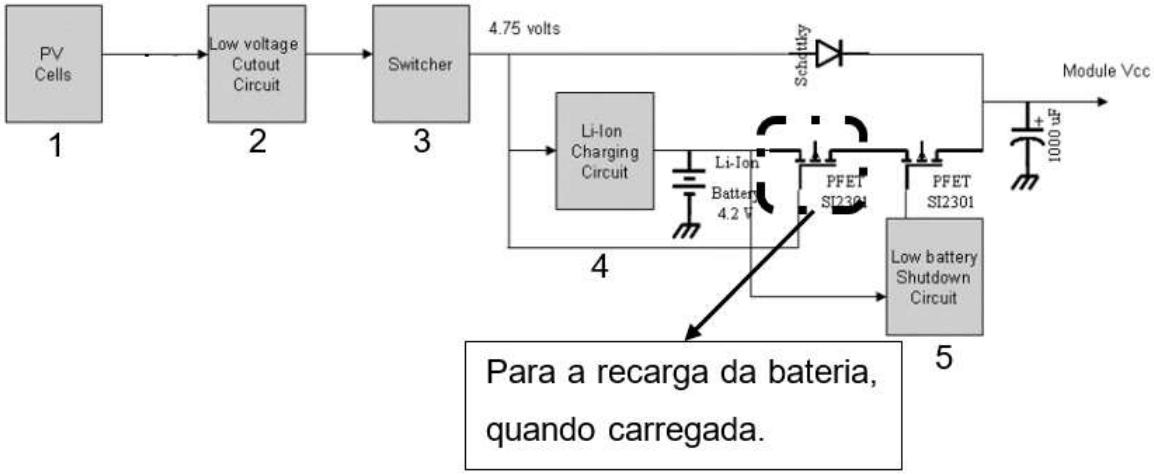
B_C → Quantidade de amperes que a bateria pode fornecer em uma hora, ou seja, a capacidade de armazenamento de energia [Ah].

I_M → Consumo médio da carga acoplada à bateria [A].

T_{SS} → Tempo sem captação de energia solar, ou seja, o tempo em que a bateria alimenta a carga [h].

Para a utilização de bateria em sistema fechado, sem alimentação externa, é necessário um elemento denominado controlador de carga, pois as baterias necessitam de um controle de carga e descarga, para que não haja nenhum incidente, como sobrecarga ou esgotamento total da bateria. Em baterias de íons lítio, existe a necessidade de desacoplar a bateria em algumas ocasiões, sendo o controlador de carga responsável por esta tarefa. O Controlador é responsável por: desacoplar a bateria da fonte carregadora quando a bateria estiver carregada, desacoplar a carga da bateria quando a bateria estiver no nível baixo e controlar as condições de carga da bateria (Figura 6) (WIN et al., 2010).

Figura 6 – Diagrama de blocos do funcionamento de um circuito para EH



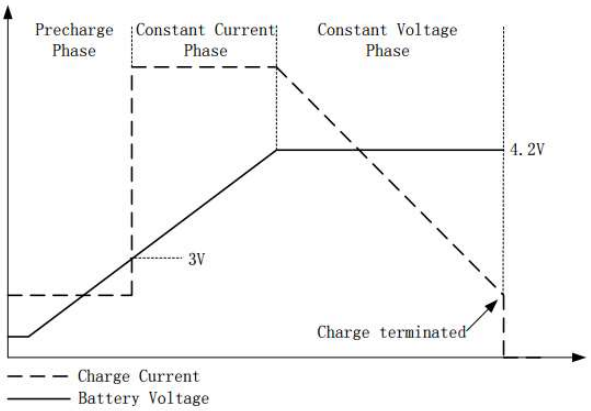
Fonte: Adaptado de WIN et al. (2010)

No circuito, os blocos de 1 a 3 permitem a coleta da energia pelo PV, que será encaminhada para o controlador de carga, que nesta etapa tem a atenção especial devido à segurança do sistema e do usuário. No bloco 4 é estabelecido as condições de carga, como tensão de 4.2V em baterias de lítio e o bloco 5 desacopla a carga da bateria quando a bateria atingi um nível baixo de tensão (Figura 6).

A carga de uma bateria tem comportamento variável, não é possível por exemplo realizar a carga instantânea.

Curva de carga é o comportamento da corrente e tensão de carga de uma bateria, na qual tem sua característica de acordo com o material. A curva de carga característica de uma bateria de íons lítio, conforme o *data sheet* do circuito integrado CN3063 (um microchip) que atua como controlador de carga, desenvolvido pela CONSONANCE (s.d.), é apresentada na Figura 7. É possível notar que existem vários regimes de carga em uma bateria, mas nenhuma é instantânea. A captura de energia mais rápida existente até o momento é a captada por Super Capacitores, porém não serão abordados pois não serão utilizados neste trabalho(ABBEY e JOOS, 2007).

Figura 7 – Curva de carga de uma bateria de lítio



Fonte: CONSONANCE (s.d.)

Observa-se na Figura 7 tem três períodos bem definidos: a pré-carga, que carrega a bateria com uma corrente constante em nível mais baixo, até a bateria atingir 3 V; após este ponto, é injetada a corrente máxima até a bateria atingir 4.2 V; e, em uma fase final vai diminuindo a corrente com uma tensão constante.

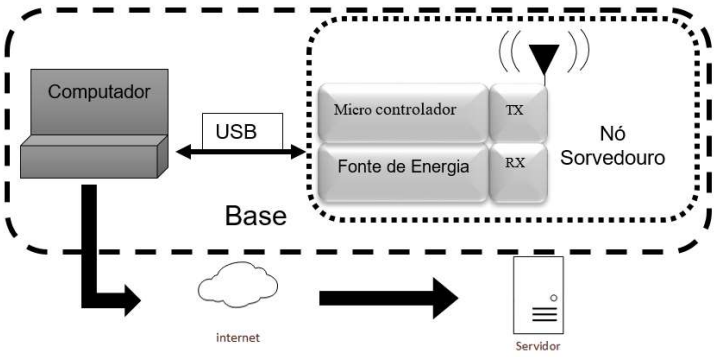
2.4 RSSF

Uma rede é um meio que conecta elementos entre si. No caso de RSSF, uma rede composta por vários nós sensores com a finalidade de monitorar grandezas específicas e comunicação entre si por meio de rádio frequência (RUIZ et al., 2004).

A RSSF é composta por dois elementos: um dos elementos é denominado nó sorvedouro, é o elemento no qual os dados escoam e são armazenados, em servidor local ou online; e, o outro é o nó sensor que fica distribuído no ambiente conforme interesse da aplicação.

A Base é responsável pelo agrupamento dos dados recebidos dos nós sensores. Este elemento pode ser o gerente de roteamento, gerente da rede, *gateway* com a internet ou mesmo ser um servidor (Figura 8).

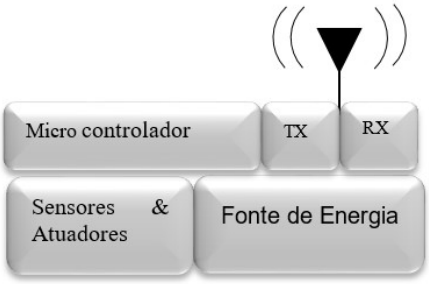
Figura 8 – Representação da base



Observar-se na Figura 8 que entre o micro controlador e o computador existe uma conexão Universal Serial Bus (USB) nos dois sentidos, uma vez que os pacotes chegam via rádio, são passados ao computador e vice-versa. O computador processa os pacotes, sendo responsável pela aquisição dos dados.

Nó sensor tem a função de medir grandezas, por meio de sensores, e enviá-los à base, sendo também atribuído em alguns casos a função de roteador ou repetidor (Figura 9).

Figura 9 – Nó sensor

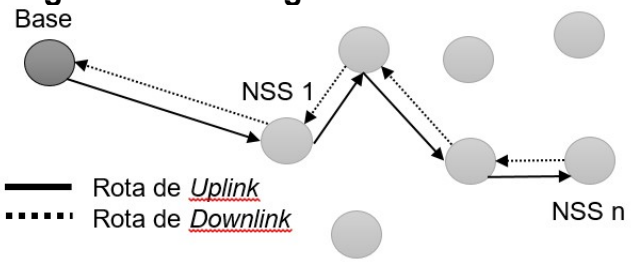


Com as Figura 8 e Figura 9, é possível avaliar a estrutura de hardware dos elementos da rede, que basicamente é composta por um Micro Controlado (MCU), transceptor, fonte de energia e sensores.

Uma visão da rede, com a disposição dos elementos e uma possível estratégia de tráfego (Figura 10) mostra como ocorre o tráfego de informação de um dos nós sensores para a base, o protocolo de comunicação e roteamento pode ser

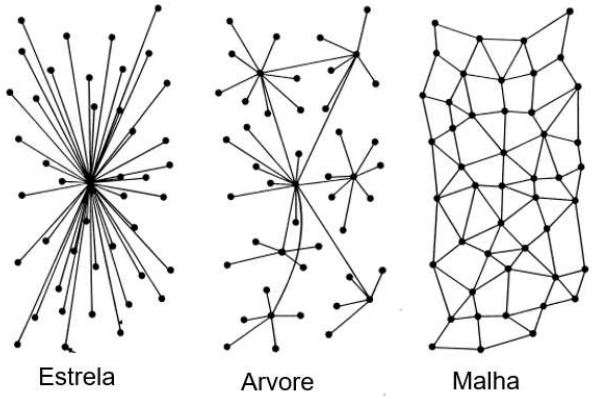
implementado conforme as necessidades da própria rede; no caso, a base é o agente que solicita.

Figura 10 – Trafego de dados em uma RSSF



As RSSF podem ter diferentes topologias de redes: estrela (centralizada) onde todos os elementos estão diretamente conectados com um único centro de dados, no caso a base; arvore (descentralizada) existem centros (bases) espalhados e interligados, na qual cada centro tem seus elementos; malha (Distribuída) não possui centros (Figura 11).

Figura 11 – Topologias de rede



Fonte: Adaptada de MARIANA LETTI (2013)

Dentre as topologias, tem os métodos com que os elementos podem se comunicar entre si, a seguir está alguns, começando com os de acesso baseados em contenção (sem ordenamento de acesso): protocolo ALOHA possui dois canais de rádio independentes, sendo um para a comunicação da base com os NS e outro o seu inverso. A colisão existe no canal NS/Base, os NS transmitem em tempo aleatório

para amenizar as colisões(BACCELLI et al., 2006); CSMA (*Carrier-Sense Multiple Access*) antes do NS transmitir, verifica se não há nenhuma transmissão sendo feita no canal; caso haja, o NS aguarda um tempo aleatório e tenta novamente.

Métodos de acesso baseados sem contenção (com ordenamento de acesso): POLLING o NS só envia pacotes quando recebe uma requisição da base. Quando existe mais de um NS, ele só responde quando o ID de destino for o dele; múltiplo salto o NS recebe o pacote, verifica se ele é o ID de destino; se for responde; caso contrário, ele decrementa um salto e retransmite até que a mensagem chegue até ao destinatário.

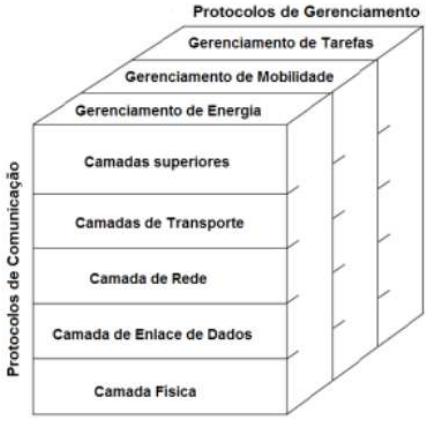
As vantagens do uso de RSSF são a escalabilidade e a ausência da necessidade de infraestrutura, tornando fácil sua aplicação. As características de uma RSSF são tolerância a erros, capacidade de se organizar e a necessidade de curto tempo de desenvolvimento (NALLUSAMY e DURAISWAMY, 2011).

As faixas utilizadas em RSSF são denominadas como bandas de ISM (Industrial, Scientific and Medical), que são bandas não licenciadas, que não necessitam de autorização da ANATEL para operação (DELGADO GOMES et al., 2012), mas cujos equipamentos devem ser certificados por ela, a banda mais utilizada para RSSF no Brasil é a banda de 915MHz, que vai de 902-907,5 e 915-928 MHz.

2.4.1 Pilha de protocolo

Como as RSSF são redes personalizadas, desenvolvidas para atender uma necessidade, ela pode sofrer adaptações. A maioria das RSSF utiliza um protocolo de 5 camadas, segundo especificações do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE). O padrão para RSSF fica na especificação IEEE 802.15.4(SOUSA e LOPES, 2011), que define as 5 camadas(Figura 12).

Figura 12 – Camadas segundo IEEE 802.15.4



Fonte: SOUSA e LOPES (2011)

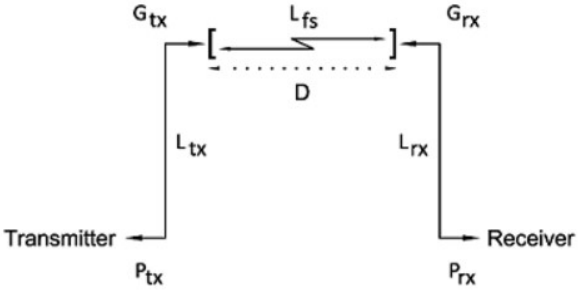
Dentre as 5 camadas do protocolo de comunicação, a camada MAC é uma das atribuições da camada de Enlace; sendo as 5 camadas tem funções distintas, para a comunicação entre elementos em uma rede 802,15,4. E os protocolos de gerenciamento são informações contidas dentro das camadas, que são úteis apenas para o gerente da rede.

2.4.2 Link budget

Através do *link budget* é possível determinar qual a potência necessária para um link, ou mesmo a sensibilidade necessária para estabelecer a comunicação via rádio.

O *Link budget* determina o alcance de um link rádio, em uma comunicação com interface aérea. Para os realizar cálculos de alcance do link, são considerados os ganhos e as perdas da comunicação (Figura 13).

Figura 13 – *Link budget*



Fonte: “Link Budget Calculator” (s.d.)

A Equação 2.2 calcula a potência recebida, considerando o balanço de ganhos e perdas da comunicação:

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} - L_M + G_{RX} - L_{RX} \quad (2.2)$$

Onde:

P_{RX} → Potência recebida [dBm]

P_{TX} → Potência transmitida [dBm]

G_{TX} → Ganho da antena de transmissão [dBi]

L_{TX} → Perda de transmissão [dB]

L_{FS} → Perda no espaço livre [dB]

L_M → Perdas sistemática [dB]

G_{RX} → Ganho da antena de recepção [dBi]

L_{RX} → Perda de recepção [dB]

Para o cálculo da perda no espaço livre utiliza-se a Equação 2.3:

$$L_{FS} = 10log_{10}(\frac{4\pi D}{\lambda})^2 \quad (2.3)$$

Onde:

D → Distância entre as antenas [m]

λ → Comprimento de onda [m]

2.4.3 Antena

Um fator importante na comunicação com interface aérea é a antena. A antena delimita a área de comunicação, como demonstrado nos cálculos do *link budget.*

Existem diferentes tipos de antenas, com áreas de coberturas diferentes. Dependendo da aplicação, não é interessante uma antena que cubra todas as direções e sim um

setor ou uma direção. A diminuição da área de cobertura de sinal de uma antena é compensada pelo ganho da intensidade do sinal em uma direção.

Um dos parâmetros, normalmente usado para avaliar as antenas, é a perda de retorno, conhecido como S11, que informa o quanto do sinal retorna do sinal injetado na antena. Outro parâmetro é o diagrama de radiação, que mostra em dois planos a cobertura da antena e o ganho.

Basicamente, existe uma antena referência e três tipos de antenas reais:

- Isotrópica (referência) – irradia para todas as direções;
- Omnidirecional – irradia em todas as direções no plano;
- Setorial – irradia apenas em um setor dos dois planos;
- Direcional – irradia em uma direção.

2.5 RSSF MONITORANDO AMBIENTE

Uma pesquisa desenvolvida na *Great Duck Island* localizada no *Gulf of Maine* na américa do Norte, espalhou 32 NS pela ilha (MAINWARING et al., [s.d.]). Para analisar o comportamento da rede no mundo real sem painel solar, desenvolveram o NS com uma consiste de uma única placa com componentes miniaturizados.

Os dados dos sensores usados nas coletas foram temperatura, pressão barométrica e umidade; apresentando um consumo menor que 1 mA por sensor. Como a única fonte de energia dos NS eram baterias, foi realizado um cálculo, denominado *Energy budget*, para que o sensor tivesse um tempo de vida de 6 meses.

O consumo do NS foi a soma do transceptor (20.000 nAh em TX e 8.000 nAh em RX) e do microprocessador (com o consumo maior de 83.333 nAh usando a memória flash e 0.011 com ADC).

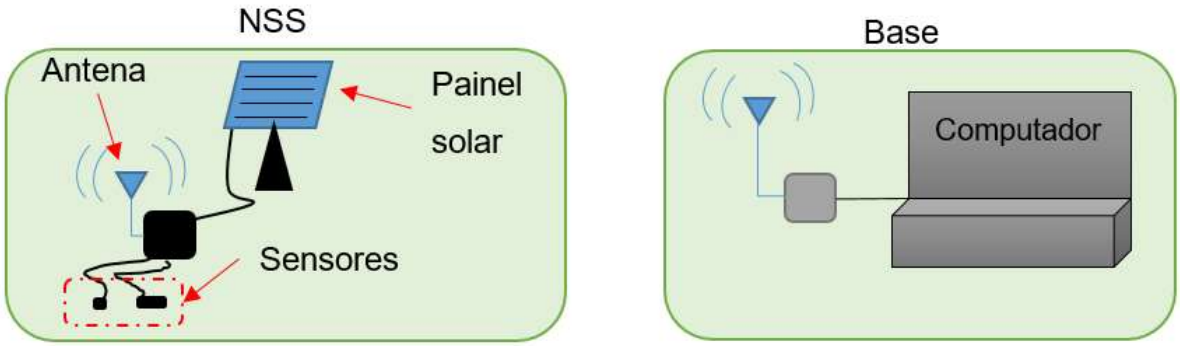
O *Energy budget* resultou dos cálculos a bateria usada, que foi um par de pilhas convencionais AA (2100 à 2800mAh (“Corrente de Pilhas Comuns (DUV373)”, [s.d.])). Os dados foram dispostos a biologistas para auxiliar em pesquisas realizadas na ilha (MAINWARING et al., [s.d.]).

As medidas coletadas ao longo do tempo se tornam dados históricos, facilitando a compreensão do comportamento do ambiente.

3 RSSF COM NÓ SENSOR SOLAR

Um NSS é um NS com uma autonomia energética através de EH com PV e bateria. A Figura 14 ilustra uma RSSF, com NSS comunicando-se com a base. O NSS deve estar disposto no ambiente dentro de um raio que possibilite um link de comunicação via rádio. Neste caso a solução de EH utilizada é solar, sendo aplicado apenas em ambiente *outdoor*, pois é necessária a incidência solar no PV. Na etapa construtiva do NSS é necessário selecionar os transdutores, sendo estes correspondentes a cada aplicação. O requerente das coletas especifica a grandeza e a precisão das mesmas e tempo entre coletas. O responsável pelo desenvolvimento da RSSF verifica se existe o transdutor que atenda à necessidade do requerente.

Figura 14 – Visão Geral de uma RSSF com NSS



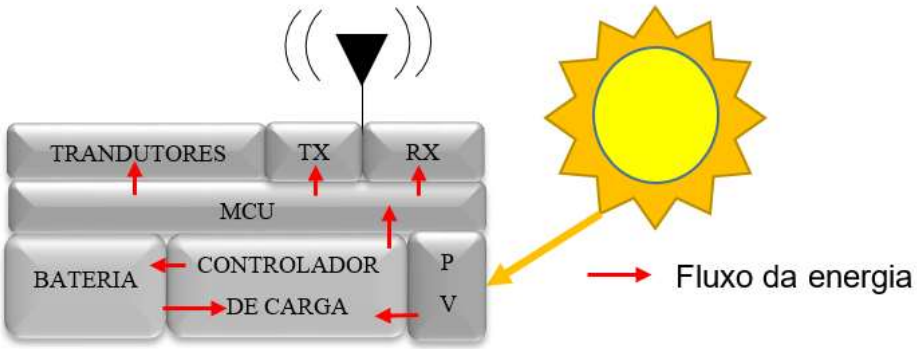
Os passos para o desenvolvimento da RSSF com nó sensor solar são: o primeiro passo é dimensionar o PV e a bateria, para que suportem a aplicação; segundo passo, certificar que o link rádio tenha uma conexão entre os elementos; Terceiro passo, fazer com que as informações coletadas cheguem ao requisitante.

Não existe um NSS que sirva para todas as aplicações, devendo ser personalizado. Porém, em se tratando de um elemento que tem uma fonte energética limitada, onde todas as aplicações demandam energia, faz com que a fonte energética seja um dos elementos mais importantes do NSS (AKYILDIZ et al., 2002). Uma otimização de gasto de energia é um fator importante e se aplica na maioria das aplicações deste tipo.

Como contribuição para a economia de energia, uma estratégia na Controle de Acesso ao Meio (MAC) que possibilite ao NSS uma permanência maior em seu estado

de dormência (menor consumo). Para melhor compreensão na Figura 15 o NSS está dividido em blocos e como ocorre o fluxo de energia entre os blocos.

Figura 15 – NSS



3.1 CONCEPÇÃO SISTÊMICA

Uma RSSF tem a necessidade de um gerente de dados e de um gerente da rede. A necessidade de dois gerentes é devido a funções bem delimitadas das informações coletadas, referentes à rede: RSSI, nível da bateria, perda de pacotes, sendo estas algumas usadas, podendo ser requeridos mais parâmetros como métricas de LQI (*Link Quality Indicator*), GPS (*Global System Position*) entre outras. Estas informações são úteis apenas para o gerente da rede, pois é vital que os NS estejam todos se comunicando e com energia suficiente para não comprometer a vida da rede.

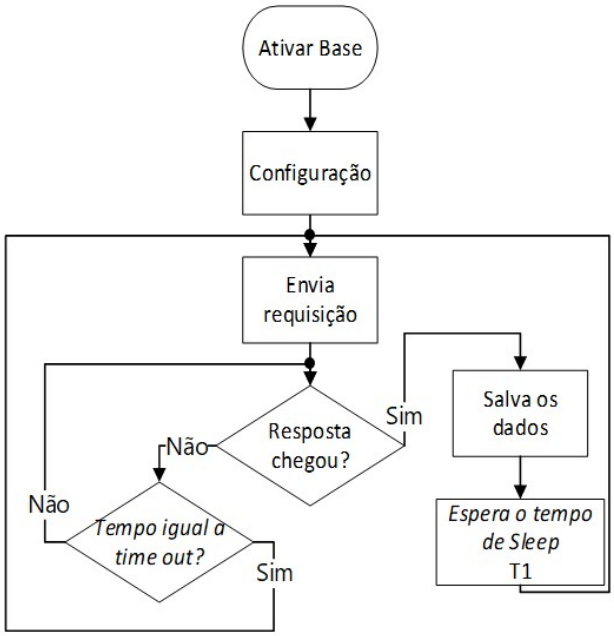
Informações referentes à aplicação: temperatura, umidade, sendo estas algumas das grandezas usadas, podendo ser adicionados outros transdutores como, barômetro, acelerômetro, pluviômetro, entre outros. O gerente de dados fica responsável por verificar a integridade dos dados coletados, e envio para especialistas da área.

3.2 CAMADA MAC PROPOSTA PARA O NSS

A MAC está na camada de Enlace (RIBEIRO, 2010) no modelo TCP/IP e é responsável pelo acesso ao meio. A proposta é atender alguns requisitos proposto, que são: o NSS responde a uma requisição; caso a base não receba a resposta da requisição, persiste até uma resposta; o NSS tem uma janela temporal para ficar em modo RX; o sincronismo da rede é responsabilidade da base (Cadência da transmissão).

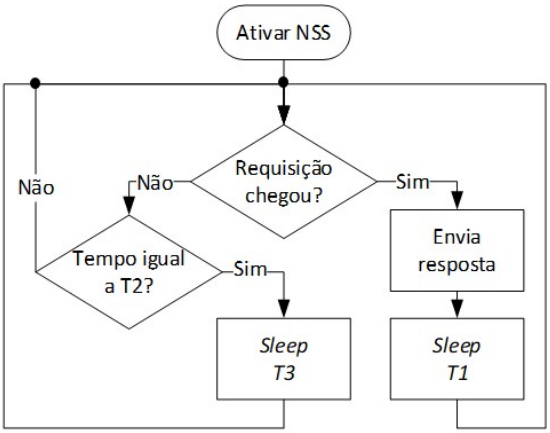
Com a proposta na camada MAC, o cenário está delimitado a um NSS e uma base, podendo ser escalável com o desenvolvimento de novos protocolos de comunicações. A implementação na base da MAC é distribuída no software do PC e no firmware do módulo rádio. O fluxograma da MAC na base (Figura 16) mostra que à Base cabe fazer a requisição e esperar um tempo para que o NSS responda. Caso o NSS não responda, a base insiste até obter uma resposta. Quando a base recebe a resposta, aguarda um período de *sleep* de T1. O protocolo do NSS tem que responder à base.

Figura 16 – Fluxograma da MAC na Base



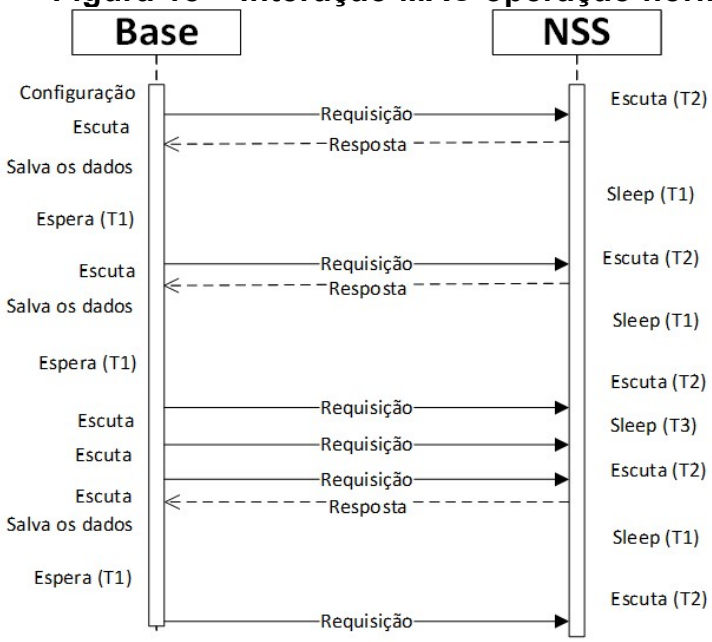
O fluxograma da MAC no NSS (Figura 17)nos mostra que no NSS existem três tempos: (T1) tempo de *sleep*, (T2) Tempo tem que o NSS fica no modo de RX e (T3) tempo em que o NSS entra em *sleep* quando não recebe uma requisição, fazendo com que o NSS use menos energia possível.

Figura 17 – Fluxograma da MAC no NSS



Na interação entre base e sensor, existe uma dinâmica representada em 3 estados: o primeiro mostra o funcionamento normal (Figura 18); o segundo, quando a base para de se comunicar (Figura 19); e o terceiro, quando o NSS para de se comunicar (Figura 20).

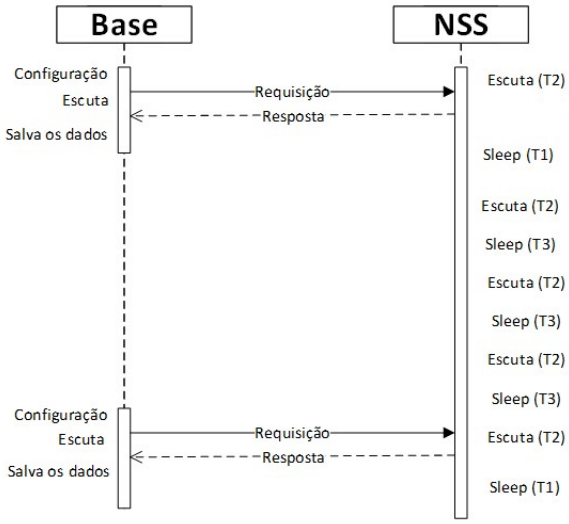
Figura 18 – Interação MAC operação normal



Na interação entre a base e o NSS não existe uma sincronia, surgindo lacunas ao longo do tempo. O tempo T3 (Figura 19) é uma estratégia para economizar energia,

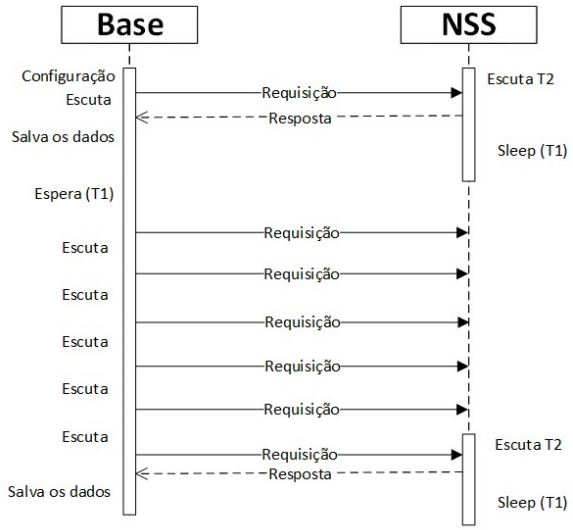
quando existir uma falha na comunicação. Quando a comunicação falha, pode ser ocasionada pela falha da base ou do NSS. Quando a base para de se comunicar, o NSS começa a alternar seu estado em RX e *sleep,* assim que a base volta a operar recupera-se a normalidade do funcionamento.

Figura 19 – Interação MAC com falha na base



O terceiro estado é quando o NSS para de se comunicar, fazendo com que a base entre em um loop de requisições até obter uma resposta (Figura 20).

Figura 20 – Interação MAC com falha no NSS

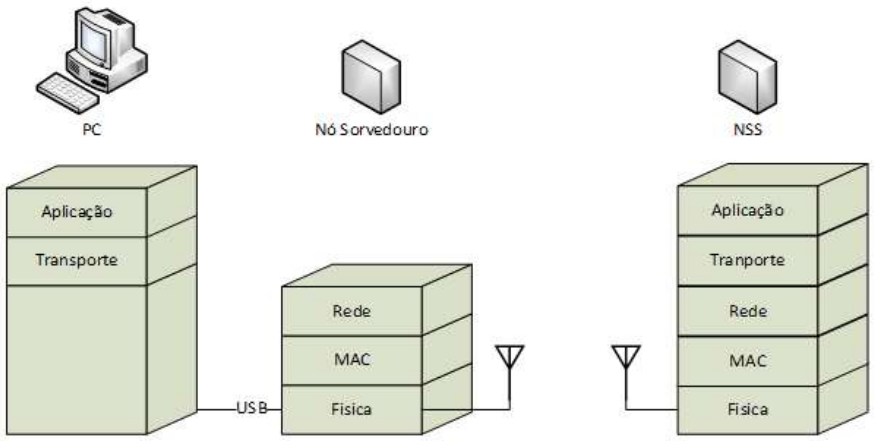


3.3 SOFTWARE DO GERENTE

O sistema, em sua parte lógica, consiste na integração de três *hardwares*. No ponto de vista do protocolo de comunicação, a Figura 21 mostra a integração dos três elementos. O computador (primeiro elemento) com maior poder de processamento, junto com o Nó Sorvedouro (segundo elemento) conectado via USB no PC, comunica-se com o NSS (terceiro elemento) através de uma interface rádio.

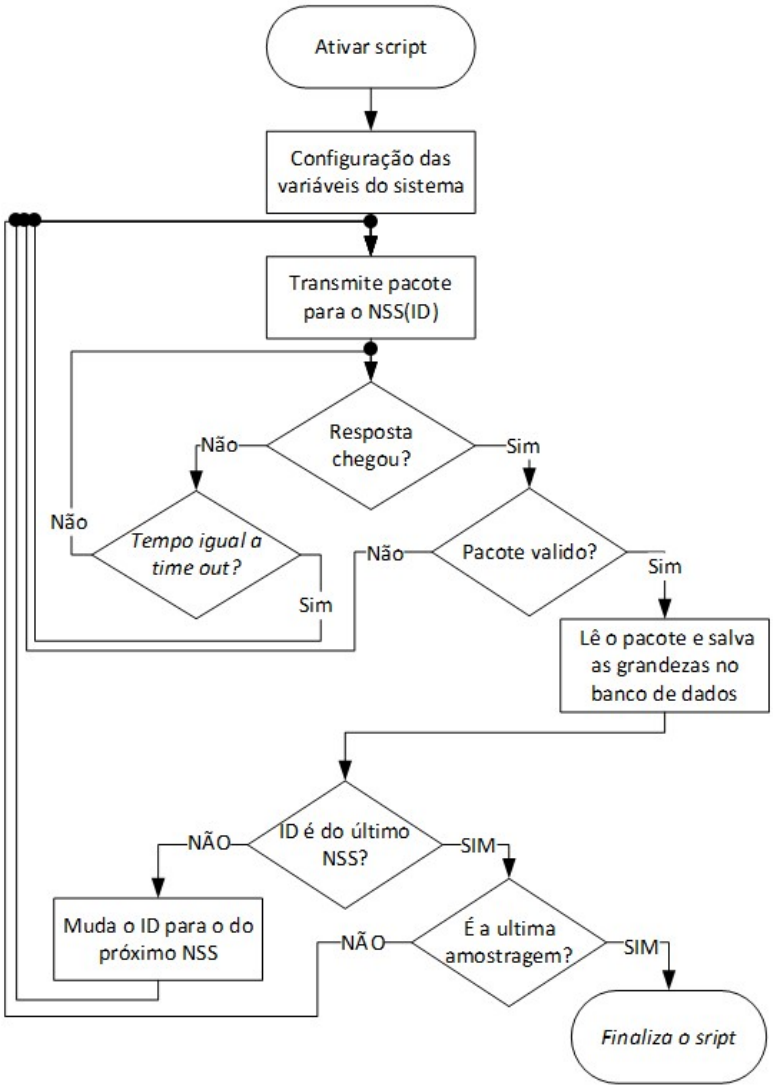
Como o computador é o elemento com o poder de processamento maior, é um consenso centralizar no computador a maior parte do processamento. Fazendo com que os demais Nós sejam mais econômicos, financeiramente e energeticamente.

Figura 21 – Protocolo de comunicação do sistema



Na Figura 22 é mostrado o fluxograma do script que roda no computador, responsável pela lógica da base, com o script em linguagem de programação Python. No fluxograma contem a estrutura da lógica que, é responsável pela gerencia da rede e armazenamento dos dados. No computador é possível salvar os dados localmente ou em um servidor na web.

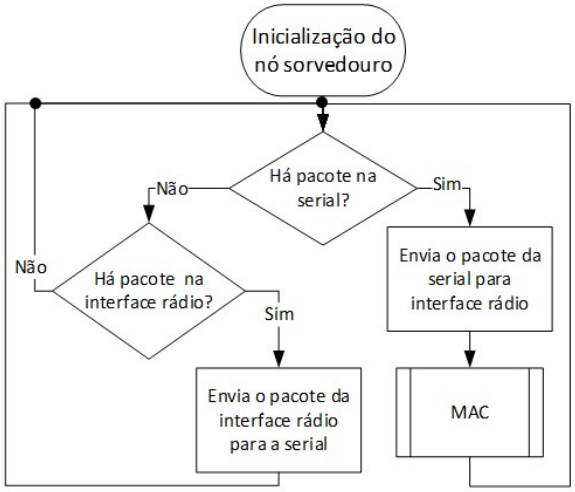
Figura 22 – Fluxograma do script em Python



3.4 FIRMWARE DA RSSF

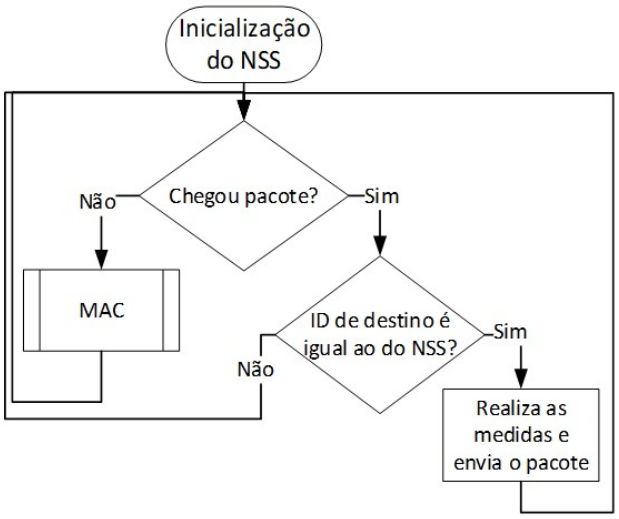
O Nó Sorvedouro tem um firmware, cuja função é deixá-lo transparente, retransmitindo o que vem pela serial para a interface rádio e vice-versa (Figura 23). O Nó Sorvedouro é praticamente um adaptador.

Figura 23 – Fluxograma do firmware do Nó Sorvedouro



O NSS carrega o firmware responsável pelo funcionamento do mesmo (Figura 24).

Figura 24 – Fluxograma do firmware do NSS



Neste caso em especifico não há o protocolo de roteamento, mas com mais sensores há a necessidade do desenvolvimento do protocolo para este fim implementado no firmware.

3.5 CÁLCULO DE ENERGIA

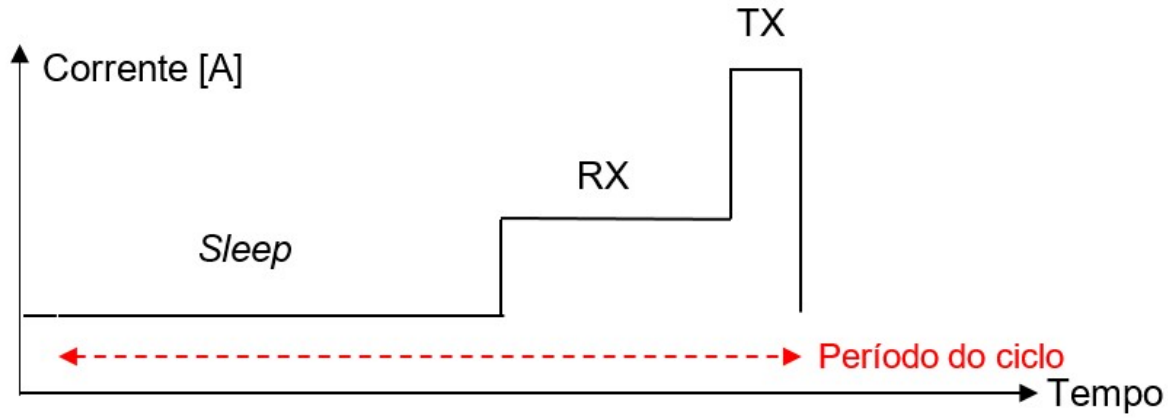
Para o desenvolvimento do NS é necessário conhecer o consumo do nó. As cargas normalmente em um nó são: micro controlador, controlador de carga, transceptor e

transdutor. Para fazer um levantamento do consumo é possível verificar as informações fornecida pelo fabricante, normalmente conhecidas como *Datasheet* no meio eletrônico, ou através de testes em bancadas.

O NSS tem três períodos existentes (Figura 25) e o equacionamento do consumo baseia-se em encontrar a corrente média de cada estado de operação de um NSS. Os três modos de operação do NSS são:

- RX aguarda uma aquisição, faz o processamento da coleta de dados;
- TX faz a transmissão dos dados e controla a carga da bateria;
- *Sleep* muda os estados dos componentes do NSS para *power down*.

Figura 25 – Modos de operação



Existe um período cíclico, que está relacionado com o intervalo de tempo quando são feitas aquisições de dados no NSS (Figura 25). A meta é deixar em *Sleep* a maior parte do tempo, porém o tempo de *Sleep* está relacionado com o tempo de amostragem, que depende da necessidade de cada aplicação. A recepção tem o tempo dependente de sincronismo da rede, pois o NSS sai do modo *Sleep* e espera a aquisição da base para executar uma rotina de amostragem. A rotina em si tem um tempo fixo, que só varia de aplicação para aplicação. Por último a transmissão, com o tempo dependendo do protocolo de comunicação e de sua taxa.

A equação da corrente média é apresentada na Equação 3.1.

$$I_{Médio} = \frac{1}{T_T}\sum_{i=1}^{3} T_i \,.\, I_i \qquad (\ 3.1\ )$$

Onde:

T_T→ Período, tempo de ciclo de aquisição de dados

T_i → Tempo em cada modo [s]

I_i→ Corrente em cada modo [A]

Na Equação 3.1 é realizada a soma das correntes usadas em cada modo de operação (RX, TX e *Sleep*), como o período é a soma dos tempos de cada ciclo, é possível encontrar o consumo médio.

A equação para o cálculo da corrente, utilizando a divisão por estados, é apresentada na Equação 3.2, adaptada da Equação 3.1.

$$I_{NS_M} = \frac{t_{TX} \times I_{TX} + t_{RX} \times I_{RX} + t_S \times I_S}{t_{TX}[seg] + t_{RX}[seg] + t_S[seg]} \qquad (\ 3.2\ )$$

Onde:

I_{NS_M}→ Corrente média do NS [A]

t_{TX} → Tempo que o modo TX fica ativo, em um ciclo [s]

I_{TX} → Corrente consumida no modo TX [A]

t_{RX} → Tempo que o modo RX fica ativo, em um ciclo [s]

I_{RX} → Corrente consumida no modo RX [A]

t_S → Tempo que o modo *Sleep* fica ativo, em um ciclo [s]

I_S → Corrente consumida no modo *Sleep* [A]

A seguir obtém-se o consumo médio de corrente, em joules (Equação 3.3). A energia necessária para um ciclo é a quantidade de joules encontrada na Equação 3.3, sabendo o tempo por ciclo é possível saber a energia necessária por hora, dia, mês.

$$E = I * T_N * 3600 \qquad (\ 3.3\ )$$

Onde:

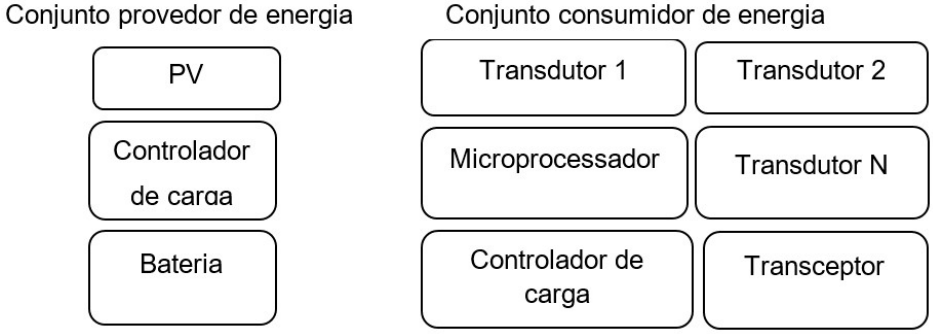
E → Energia [J]

I → Corrente média [Ah]

T_N → Tensão nominal [V]

O dimensionamento do conjunto responsável pela alimentação do NSS (controlador de carga, bateria e um PV), é feito através de um balanceamento de energia, dividido em dois conjuntos (um provedor, outro consumidor) (Figura 26). Por parte do consumidor, o intervalo de amostragem tem um impacto alto no consumo.

Figura 26 – Provedores e consumidores de energia



Para balancear esses conjuntos é necessário primeiramente determinar o consumo do conjunto consumidor, para depois, obter energia suficiente do conjunto provedor.

Em seguida, é o conjunto provedor, sendo que os três elementos têm papeis distintos: o PV é o conversor de energia, que deve ser capaz de gerar energia suficiente para alimentar o NSS, enquanto houver sol, e o montante necessário para suprir o NSS

quando não houver; o controlador ajusta a velocidade de carga da bateria, podendo se tornar um gargalo quando não considerado; o papel da bateria é suprir a ausência do sol, podendo ser dimensionado para suportar dias sem o mesmo.

Para dimensionar o PV é necessário conhecer a irradiação média diária do local de instalação, a eficiência de conversão e a energia necessária para permanecer na ausência de sol, visto que é dependente de cada aplicação.

3.5.1 Cálculo do PV

A média de incidência solar diária é encontrada em bancos de dados (CRESESB - Centro de Referência para Energia Solar e Eólica Sérgio Brito) utilizando a latitude e a longitude para determinar a irradiação média diária (DUFFIE et al., 2003). As demais informações do PV são fornecidas pelo fabricante.

A Equação 3.4 apresenta o cálculo para determinar a energia necessária para suprir a demanda.

$$E_{PV} = Ef_{PV} * I_{MD} * \acute{A}rea_{PV} * \mathrm{T} \qquad (3.4)$$

Onde:

E_{PV} → Energia que o PV converteu [J]

Ef_{PV} → Eficiência de conversão de energia do PV, energia luminosa em elétrica

I_{MD}→ Irradiação solar media ao longo do dia [W/m²]

T → Tempo [s]

$\acute{A}rea_{PV}$ → Tempo [m²]

Como o PV é constituído por células, a sua escolha deve considerar a quantidade de células aproxima da área necessária.

O controlador necessita de uma atenção para que ele não seja um limitador, pois a energia captada através do PV não é armazenada instantaneamente. Para que o controlador não seja um limitante, a corrente de carga deve atender à necessidade de

carregar o valor necessário de energia no período de sol, para suprir a demanda da carga quando não houver sol. Deve ser verificado se a bateria suporta tal corrente mínima (Equação 3.5).

$$I_C = \frac{C_B}{i_S \times E_{PV}} \quad (3.5)$$

Onde:

I_C → Corrente elétrica de carga [A]

C_B → Capacidade de armazenamento da bateria [Ah]

i_S → Incidência solar, período com luz solar [h]

E_{PV} → Energia solar coletada pelo PV [W]

Deve ser determinado o período que o NSS deve funcionar só na bateria. A capacidade é o produto do tempo e consumo (Equação 3.5 e Equação 3.6, para o equivalente em joules).

$$E_B = T_B \times C_B \times 3600 \quad (3.6)$$

Onde:

E_B → Energia da bateria [J]

T_B → Tensão nominal da bateria [V]

C_B → Capacidade de armazenamento nominal da bateria [Ah]

4 MATERIAL E MÉTODOS

4.1 BANCADA

A bancada é um *setup* para realizar medidas dos artefatos desenvolvidos, com a finalidade de automatizar coletas de dados e verificar o funcionamento dos artefatos.

- Microprocessador

O microprocessador usado é o Atemega328 AVR de 8bits, com entradas e saídas digitais e analógicas, para automatização da coleta de dados e prospecção das grandezas em bits, no caso os conversores Analógico/Digital do Microprocessador Atmega328 (ATMEL, 2010).

- Circuito Condicionador

O circuito condicionador, é necessário: (1) reduzir pela metade a tensão real, enviada ao micro controlador, pois a tensão é maior do que a porta do A/D suporta; (2) aferir a corrente do circuito, utilizando de um resistor shunt, e mensurando a corrente pela diferença de tensão antes e depois do mesmo (Equação 4.1).

$$I = (V_{Sin} - V_{Sout}) \div R_S \qquad (4.1)$$

Onde:

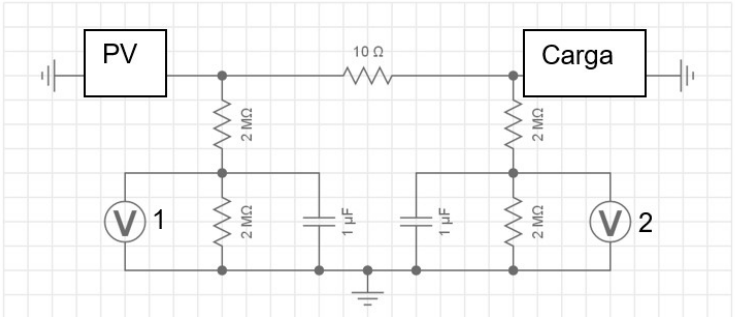
V_{Sin}= → Tensão antes do resistor Shunt [V]

V_{Sout} → Tensão depois do resistor Shunt [V]

R_S → Valor do resistor Shunt [Ω]

A Figura 27 mostra o diagrama elétrico do circuito montado em uma *protoboard*, a resistência de 10 Ω é o resistor Shunt, a resistência de 100 KΩ, a carga; os V1 e V2 representam entrada dos A/D's e do restante é o dimensionamento do circuito com suas respectivas características.

Figura 27 – Diagrama do circuito condicionador



Fonte: Através do APP *Every Circuit*

4.2 NSS E NÓ SORVEDOURO

Para a construção dos nós são necessários os seguintes itens:

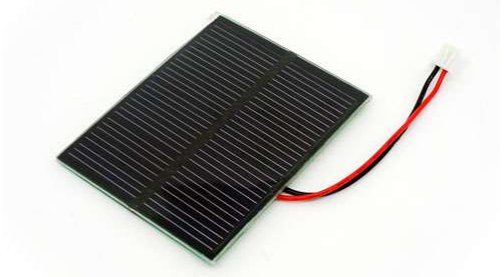
- Controlador de carga CN3063;

Utilizada uma bateria de lítio, responsável por garantir o funcionamento da bateria (CONSONANCE, [s.d.]).

- PV

Foi utilizado um PV composto por 4 células solares ligadas em paralelo. As células solares são de Silício monocristalino, com uma eficiência de conversão de 17% e uma dimensão de 55x77x3(±0,2)mm(SEEED, [s.d.]) (Figura 28).

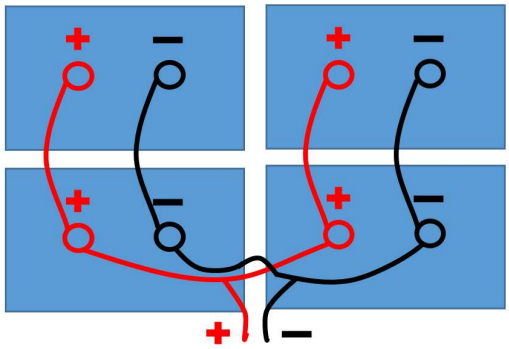
Figura 28 – Célula solar usada



(SEEED, [s.d.])

Os conectores (Figura 28) foi removido e, com os fios, o painel foi configurado (Figura 29).

Figura 29 – Configuração do PV



- Bateria

A Bateria usada é uma bateria Lítio íons, com tensão nominal de 3,7V e a capacidade de 500mAH (Figura 30).

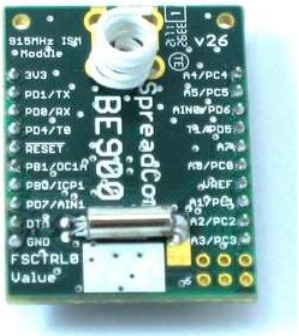
Figura 30 – Bateria usado no NSS

Fonte: (SEEED, [s.d.])

- Modulo BE900

O módulo BE900 (Figura 31) possui um processador AVR Atmega328, transceptor TI CC1101 RF, opera na banda não licenciada ISM de 915MHz; possui a certificação da ANATEL (Órgão regulamentador das telecomunicações no Brasil) e é programável através da IDE Arduino, com inúmeras bibliotecas para esta plataforma (RADIOIT, 2012).

Figura 31 – Rádio BE900



Fonte: RADIOIT (2012)

A escolha deste módulo foi por ser tecnologia nacional. O Anexo 1 apresenta o datasheet do BE900.

- Plataforma Radiuino

A plataforma Radiuino foi desenvolvida com um conceito para criar rede de sensores sem fio (RSSF) sendo uma plataforma livre.(“Home - Radiuino”, 2016). A comunicação segue um protocolo com pacote de 52 bytes (Tabela 2), sendo alguns bytes já pré-definidos e outros como TBD (*To Be Defined*), de acordo com a necessidade das diferentes aplicações.

Tabela 2 – Mapa de pacote Radiuino

Camada	Função	Byte do pacote	Camada	Função	Byte do pacote
FÍSICA	RSSI_D	0	AD_4	TYPE	28
	LQI_D	1		PART_H	29
	RSSI_U	2		PART_L	30
	LQI_U	3	AD_5	TYPE	31
MAC	SLEEP_1	4		PART_H	32
	SLEEP_2	5		PART_L	33
	SLEEP_3	6	IO_0	TYPE	34
	TBD	7		PART_H	35
NET	ID_DST	8		PART_L	36
	NID_DST	9	IO_1	TYPE	37
	ID_SRC	10		PART_H	38
	NID_SRC	11		PART_L	39
Transporte	COUNT	12	IO_2	TYPE	40
	TBD	13		PART_H	41
	TBD	14		PART_L	42
	TBD	15	IO_3	TYPE	43
AD_0	TYPE	16		PART_H	44
	PART_H	17		PART_L	45
	PART_L	18	IO_4	TYPE	46
AD_1	TYPE	19		PART_H	47
	PART_H	20		PART_L	48
	PART_L	21	IO_5	TYPE	49
AD_2	TYPE	22		PART_H	50
	PART_H	23		PART_L	51
	PART_L	24			
AD_3	TYPE	25			
	PART_H	26			
	PART_L	27			

Fonte: Adaptado de RADIOIT, 2012

Equipamentos auxiliares.

- Multímetro:

Para aferição e para verificação do funcionamento dos circuitos.

- *Vector Network Analyzer* (VNA):

Master MS2036A, para testes de antenas, em aplicações militares e civis, portável e integrado com duas entradas para testes. As entradas suportam, como VNA, um *range* de 2MHz a 6GHz e, como analisador, de espectro um range de 9KHz a 7.1GHz (ANRITSU, [s.d.]).

- Osciloscópio:

Osciloscópio digital Tektronix TPS2024B, para aferição, testes e coletas de dados.

- IDE Arduino

A IDE Arduino é a interface que possibilita a gravação do *firmware* no micro controlador e no modulo BE900. O código utiliza a linguagem C, e a atribuição dos pinos depende do hardware onde será utilizado.

4.3 MÉTODO

O método utilizado foi dividido na montagem da RSSF, e nos métodos de coleta em dois pontos. Considerando o dimensionamento do sistema energético do NSS.

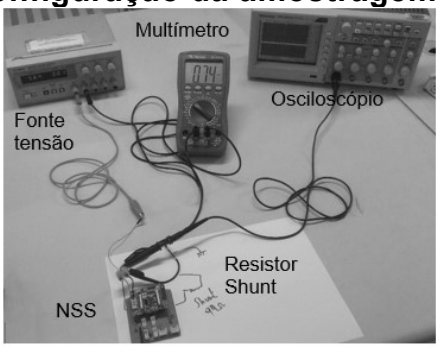
4.3.1 Montagem da RSSF

O foco do trabalho é na bateria do tipo secundária, pois, o objetivo desta dissertação é na possibilidade de armazenar a energia obtida através de EH, para ser utilizada na ausência da fonte de energia.

A Topologia escolhida foi a estrela com de *Pooling*, por ser mais simples e robusta na tratativa de problemas com colisões, sendo uma topologia centralizada através da base. Definida a plataforma, foi medido o consumo dos elementos do NSS com o multímetro ligado em série, medindo a corrente, e um osciloscópio, verificando a tensão em um resistor Shunt.

As configurações para a medida em cada modo foram realizadas através de mudança de *firmware* no NSS. O controlador de carga no teste de consumo com bateria foi alimentado por uma fonte de tensão. O Valor do Shunt usado foi de 99 Ohms, para conseguir coletar a queda de tensão no resistor, por se tratar de uma corrente muito baixa, o valor nominal do resistor é de 100 Oms, a fonte fornecia uma tensão superior 5V, devido à queda de tensão no shunt (Figura 32).

Figura 32 – Configuração da amostragem de corrente do NSS



Para auxiliar no dimensionamento do PV e da Bateria, na Tabela 3 determina-se o consumo do NSS. Para os cálculos da predição é necessário alimentar com as informações de consumo dos modos. No exemplo, está um período de amostragem de 300s correspondente a 5 minutos. Alterando os valores de *sleep* foi gerada uma segunda Tabela 4, de forma que a taxa de amostragem se alterasse.

Tabela 3 – Entrada de valores

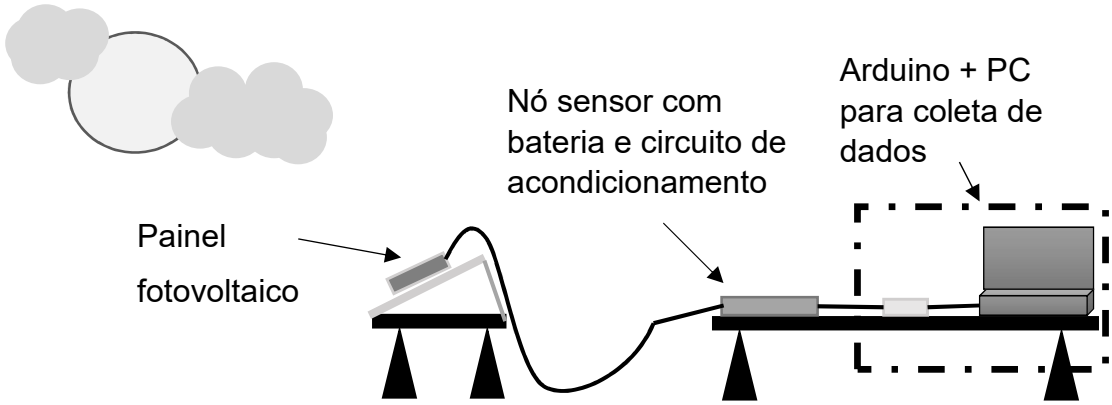
	Consumo[A]	Período[s]
Sleep	0,00012	299,488
RX	0,022	0,5
TX	0,032	0,012

Tabela 4 – Consumo de períodos por intervalos de amostragem

	1min	4min	10min	15 min	30min	1hora	2horas
Ciclo	3,09E-04	1,67E-04	1,39E-04	1,33E-04	1,26E-04	1,23E-04	1,22E-04
Hora	1,85E-02	2,51E-03	8,33E-04	5,30E-04	2,53E-04	1,23E-04	6,08E-05
Dia	4,44E-01	6,02E-02	2,00E-02	1,27E-02	6,06E-03	2,96E-03	1,46E-03
Semana	3,11E+00	4,21E-01	1,40E-01	8,91E-02	4,24E-02	2,07E-02	1,02E-02
Mês	1,33E+01	1,80E+00	6,00E-01	3,82E-01	1,82E-01	8,87E-02	4,38E-02

Para analisar o comportamento deste NSS concebeu-se uma bancada para monitorar o fluxo de energia do NSS, para fazer um *survey* do sistema em laboratório. Na bancada foi monitorado ao longo do tempo as correntes e tensões do sistema. Com os resultados é possível avaliar estratégias de dimensionamento, ferramentas de economia de energia, que permitam com que a carga tenha energia suficiente. A proposta da bancada como mostra a Figura 33 encontra-se em ambiente de laboratório, onde é monitorado o NSS.

Figura 33 – Bancada de teste



Com a bancada é possível mudar as configurações e avaliar o impacto causado na geração da energia, como a quantidade de bateria e células fotovoltaicas (PV), para fazer um dimensionamento mais coerente com o ambiente no qual será empregado o nó sensor. O range de corrente para monitorar o NSS é de 10-100 mA com precisão de 1 mA. Levando em consideração que o medidor da bancada é um microprocessador que possui AD de 10 bits, e trabalha com uma tensão referencial de 5 V, resultando em um passo de 4,8 mV (Equação 4.2).

$$V_{AD} = \frac{V_r}{2^{bits}} \qquad (4.2)$$

Onde:

V_{AD} → Passo do AD, tensão que o AD quantifica em um número digital.

$\boldsymbol{V_r}$ → Tensão de referência, Tensão que representa o máximo possível

$\boldsymbol{bits}$ → Nº de bits do AD

A corrente que passa no sistema é aferida através da queda de tensão em um resistor considerando que passará 10 mA. Usando 1 Ω obtemos uma queda de tensão de 10 mV e para 100 mA uma queda de tensão de 100 mV, com uma precisão de 4,8 mA por passo. Aumentando a resistência para 10 Ohms obtém-se uma queda de tensão de 100 mV para 10 mA e 1v para 100 mA com uma precisão de 480 µA por passo. Para melhor visualização destas informações verificar na Tabela 5. A escolha do resistor shunt foi 10 Ohms por ser múltiplo de 10 e ter uma melhor precisão comparado ao de 1 Ω, e já atende a precisão de 1 mA visto que fora erros sistemáticos a precisão é de ±0,48 mA.

Tabela 5 – Queda de tensão nos *Shunts* de acordo com a corrente

	1Ω	**10Ω**
10 mA	10 mV	100 mV
100 mA	100 mV	1 v
mA/passo	4,8	0,48

Com a bancada pronta, foi feita uma sequência de testes com configurações diferentes, podendo assim avaliar os resultados conforme a configuração muda.

Primeiramente é implementado o protocolo Radiuino no micro controlador, que é responsável pela coleta da informação de quatro tensões, com as tensões é possível obter a corrente da bateria e do PV, para isso é utilizado um código em C, carregado no micro controlador.

Com o micro controlador pronto é necessário rodar um *script* desenvolvido em Python, que requisita a captura da informação e armazena a mesma em banco de dados. Os dados são posteriormente exportados para uma planilha Excel, na qual são feitas as análises dos dados obtidos.

A bancada ficou dividida em duas partes: a primeira localizada na parte interna do laboratório LP-SiRa (Laboratório de Pesquisa em Sistema Radio) (Figura 34), com uma base para fazer aquisição para o NSS; e a segunda, na parte externa (Figura 35), mostrando o suporte para o PV, que permite alterar o ângulo da superfície das células solares.

Figura 34 – Bancada de testes

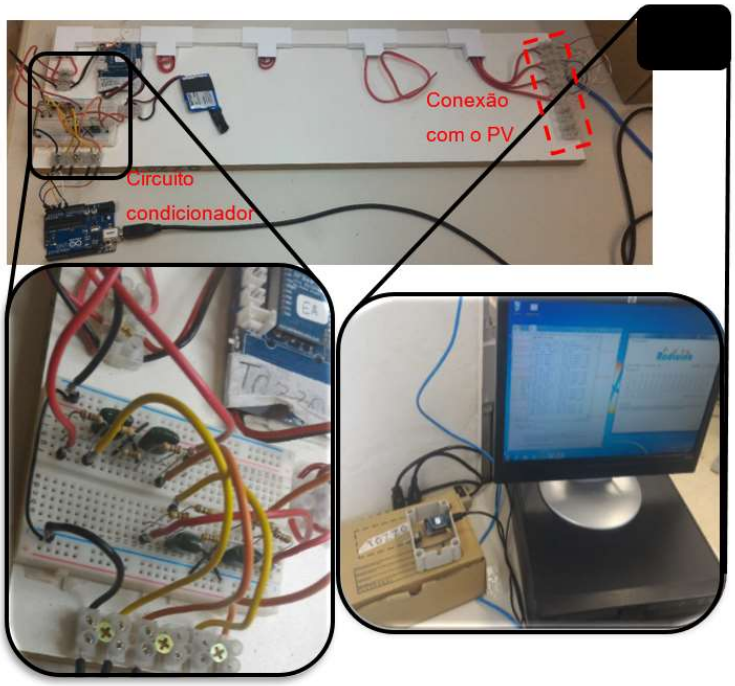


Figura 35 – Suporte para o PV

Para atender a MAC proposta foi desenvolvida uma estratégia denominada *sleep2*. Este modo de *sleep* defini três tempos T1, T2 e T3.

- T1 é o tempo equivalente do *sleep* normal;

- T2 é o tempo em que o NSS fica em RX;
- T3 é o tempo que o NSS vai ficar em *sleep* caso ele não receba nenhuma requisição quando em T2.

O *sleep2* surgiu como uma solução, para economizar energia quando existir uma falta de requisição

Para avaliar o consumo do NSS em relação à bateria, foram realizados dois testes com a bancada, amostrando a tensão da bateria: o primeiro teste, usando *sleep1*, para avaliar o tempo de descarga da bateria com três configurações diferentes:

- A primeira configuração usa um resistor como carga, a corrente de descarga é de 27 mA, uma corrente média de consumo quando não implementado o modo *sleep* no rádio;
- A segunda, com o NSS em RX o tempo todo;
- A terceira, com o NSS alternando seu modo de RX para *sleep*, sendo 4 s em RX e 15 s em *sleep* para efeito de teste.

O segundo teste foi realizado usando o *sleep2*, que permite que o NSS, quando não solicitado, alterne seu modo de RX para *sleep*, economizando energia e modificando no *sleep2* o tempo de *sleep* (T1).

- 3,1,1 (3x15=45s tempo de *sleep* T1, 15 T2e 15 T3)
- 2,1,1 (2x15=30s tempo de *sleep* T1, 15 T2e 15 T3)
- 1,1,1 (1x15=15s tempo de *sleep* T1, 15 T2e 15 T3)

O posicionamento do NSS interfere no cálculo do *link budget* usado para dimensionar as configurações de rádio frequência. Com o link estabelecido é necessário aplicar a topologia da rede, que no caso é *pooling*. Para executar o *pooling* e coletar os dados foi desenvolvido um *script* em Python na base. Na Figura 22 foi apresentado o fluxograma de como funciona o script e no Item Apêndice 1 está o código inteiro.

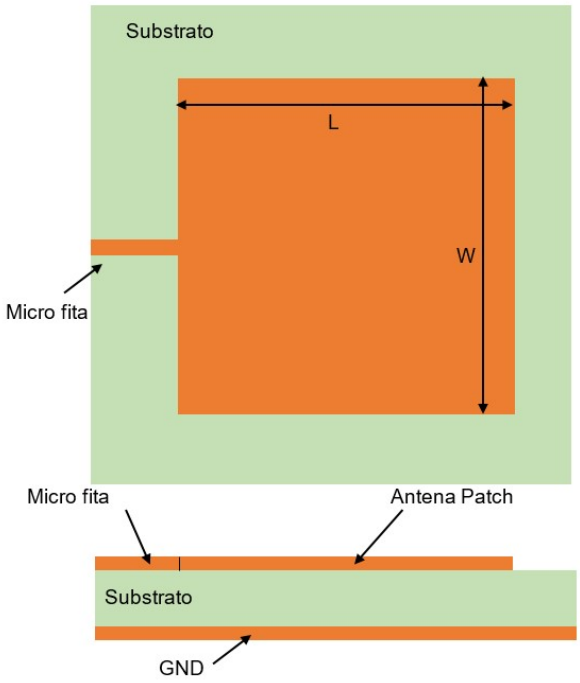
O firmware desenvolvido para o nó sorvedouro encontra-se no Apêndice 3.

O firmware do NSS é alterado conforme a aplicação, sendo atribuído ao firmware do NSS a interação com os transdutores e em alguns casos roteamento de dados. Os dados salvos pelo *script* ficam armazenados no computador em um arquivo com extensão .db, gerado pelo SQLite. O SQLite é um banco de dados Open Source embutido, que não necessita de um sistema gerenciador de banco de dados.

Para a montagem da rede definem os pontos dos NSS. Quando existir mais de um NSS é necessário estabelecer a topologia da rede. Caso existam NSS fora do raio de abrangência da base, deve-se utilizar NSS como repetidores para criar um link com a base. Um procedimento importante é o teste *survey* do local, que analisa se realmente os NSS estão se comunicando com a base.

Neste trabalho foram utilizadas antenas setoriais. A antena do NSS foi confeccionada, com uma placa de fibra de vidro 10x10cm, usando a técnica de construção de antenas de micro fita. A antena é do tipo patch retangular com duas seções L e W e uma micro fita que conecta com o gerador de sinal (Figura 36).

Figura 36 – Dimensões da antena de micro fita



Fonte: Adaptada de “Microstrip Antennas: The Patch Antenna”, [s.d.]

Para realizar o calculo das duas seções foram usadas as Equações 4.3 e 4.4 (LUCAS; SILVA, 2015).

$$f_C = \frac{C}{2L\sqrt{\varepsilon_r}} \qquad (4.3)$$

Onde:

f_C → Frequencia de corte

C → Velocidade da luz

ε_r → Constante dieletrica

$$\varepsilon_{reff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}(1 + 12\frac{h}{W}) \qquad (\ 4.4\ )$$

Onde:

ε_{reff} → Constante dieletrica efetiva

C → Velocidade da luz

ε_r → Constante dieletrica

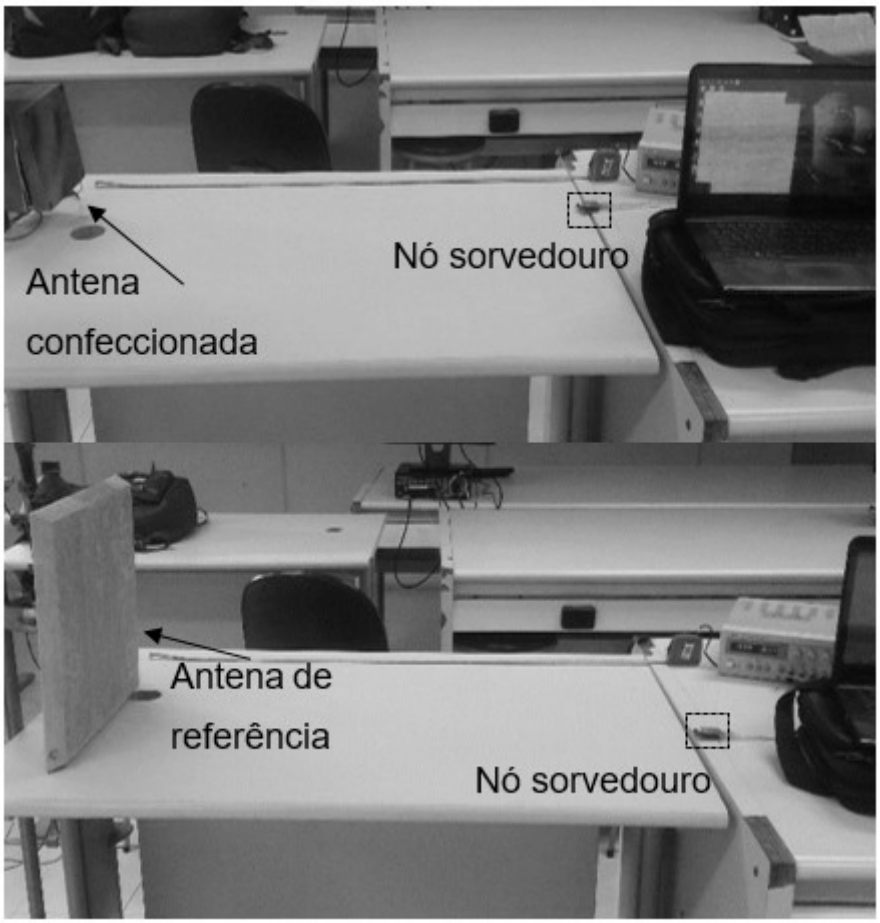
h → Espessura do dieletrico

A frequência de corte usada foi a 915 MHz, espessura da placa de 1,5 mm e o dielétrico da fibra de vidro de 4.4. Para o cálculo foi utilizado um site (“Microstrip Patch Antenna Calculator”, [s.d.]). Após confeccionada passou por ajuste fino, ajuste de sintonização (diminuição da área da antena caso o parâmetro S11 esteja acima do desejado e o inverso caso contrário).

Para verificar o desempenho da antena é utilizado o parâmetro S11, que mede a perda de retorno e indica o casamento de impedância da antena. A antena foi conectada ao VNA e configurado, para que o VNA analisasse a banda de frequência do funcionamento da antena.

O ganho da antena foi mensurado através de uma antena referência. O teste foi realizado com a distância de um metro entre a antena de transmissão e recepção. Foi medido a RSSI de uma antena de 12,5dBi e comparada com a antena confeccionada. O teste foi realizado no laboratório da PUC Campinas, com o NSS alimentado por uma fonte de tensão e os dados coletados por um computador, a configuração usada está apresentada na Figura 37.

Figura 37 – Teste de ganho da antena



Na Figura 37 a única mudança que houve foi a substituição das antenas, após a coleta de 50 amostras com cada uma.

4.3.2 Ponto de coleta 1

Para fins de teste, foram realizadas coletas de temperatura e umidade do ar. As medidas foram realizadas no campus 1 da PUC-Campinas, próximo a um espelho d'agua (Figura 39). Foram configuradas amostras a cada 4 min. Os dados mostram a data e a hora de coleta, em arquivo Excel. A distância entre o ponto de coleta e a base de coleta é de aproximadamente 15 metros em linha de visada (Figura 38).

Figura 38 – Distância entre as antenas do NSS e da base

Fonte: (“Google Maps”, 2017)

O transdutor usado foi o DHT22 que, em um único elemento, coleta a temperatura e umidade do ar. O consumo do DHT22 é de 1,5 mA segundo o fabricante (LIU, [s.d.]). Este consumo deve ser acrescentado no consumo do NSS.

Foi realizado um survey no local, analisando a corrente de carga e a RSSI do sinal. A corrente de carga foi coletada com um multímetro, obtendo uma corrente de 80mA com o céu nublado e 166mA com sol direto. A RSSI foi avaliada pelo próprio software, medindo 65dBm *down link e up link*.

Figura 39 – NSS no ponto de coleta 1

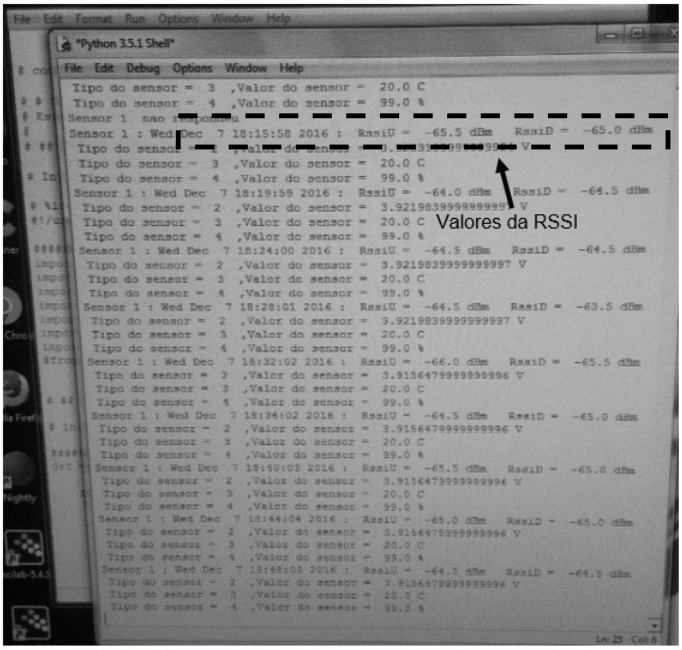
O valor do consumo na Figura 39 foi obtido pela análise da corrente fornecida da bateria para a carga, com o PV desacoplado. Para analisar a “colheita de energia”, é

medida a corrente fornecidas pelo PV. No local foi feito um survey das correntes fornecida pelo PV com o céu nublado e com o sol aberto (Figura 40).

Figura 40 – Corrente com o tempo nublado e ensolarado

O *survey* do consumo, foi realizado através de medidas empíricas. Para a gerência da rede é importante a conexão, sendo avaliada a RSSI apenas com o intuito de certificar que existia a conexão rádio, sem avaliar a qualidade da mesma (Figura 41).

Figura 41 – *Survey* da RSSI

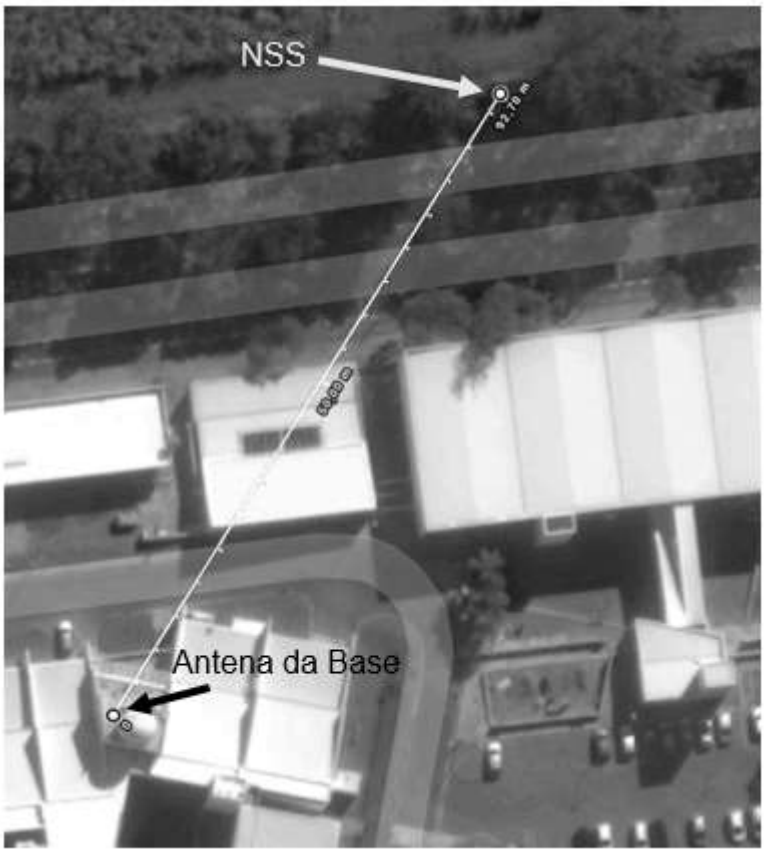


Com as informações sobre o local, foi feito o cálculo do consumo de energia, para o dimensionamento da geração e armazenagem de energia. Os dados coletados são disponibilizados através de logs, no banco de dados SQLite.

4.3.3 Ponto de coleta 2

O outro teste foi realizado com uma maior distância e em um ambiente que se aproxima de uma aplicação real. Algumas alterações na constituição física do NSS foram necessárias na parte mecânica do NSS. O local escolhido foi um ambiente próximo a um recorte de mata, com a distância aproximada de 93 m, na divisa da PUC Campinas (Figura 42).

Figura 42 – Local de coleta do NSS



Fonte: “Google Maps”(2017)

A estrutura do NSS foi reforçada com um cano metálico e camuflado, para que o mesmo se integrasse ao meio (Figura 43).

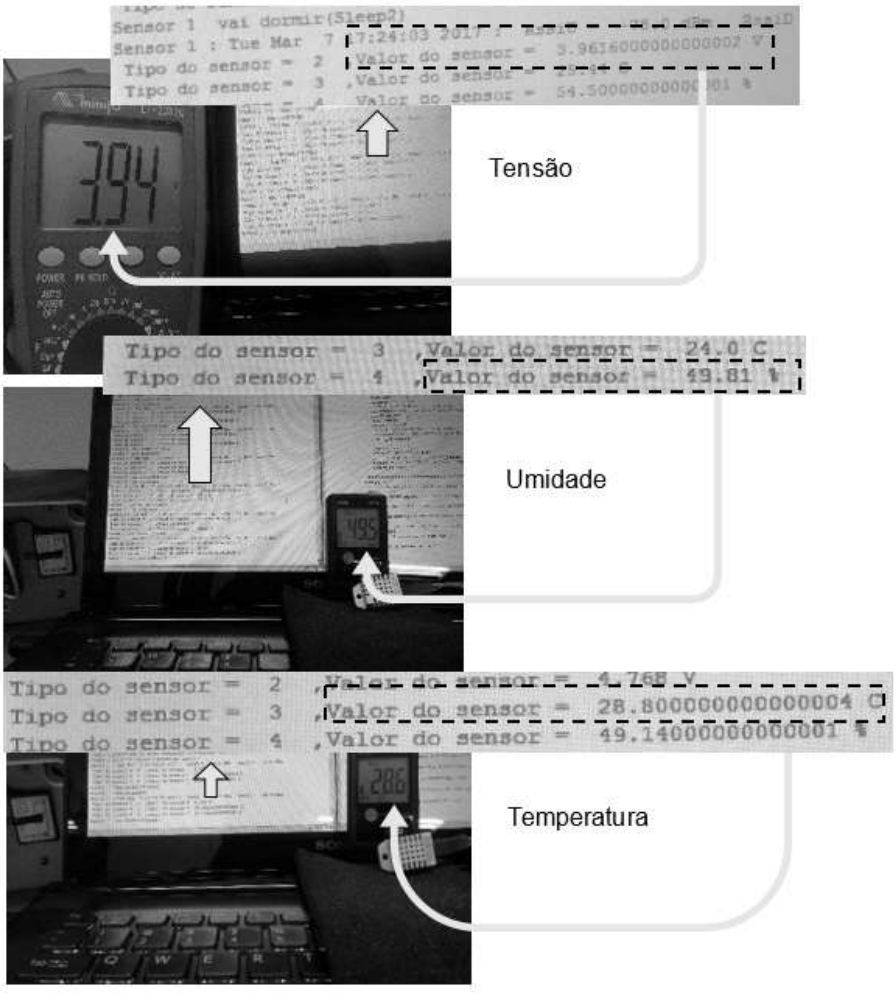
Figura 43 – NSS no local da coleta 2

Um ponto a se destacar na Figura 43 a camuflagem do NSS fez com que o mesmo se confundisse com a vegetação, demostrando ser conveniente esse tipo de precaução, para a preservação do NSS.

As configurações também foram modificadas, como a taxa de transmissão que caiu de 38 kbps para 4,8 Kbps, aumentando o alcance do link. Como a periodicidade de coleta dos dados era baixa, não houve alteração no processo de medida. Os transdutores foram aferidos com equipamentos auxiliares (Figura 44).

Sendo observados os valores próximos, com uma pequena margem de erro.

Figura 44 – Calibração dos Transdutores



5 RESULTADOS E ANÁLISE DOS DADOS

5.1 MONTAGEM DA RSSF

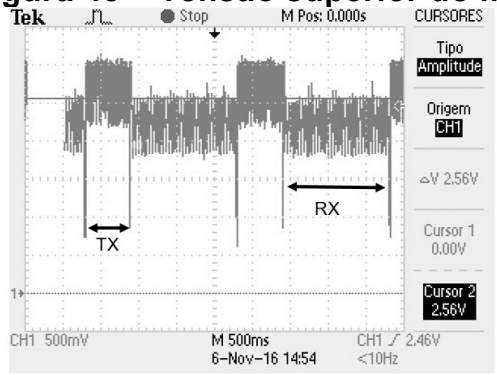
Dividindo o consumo em *seus respectivos modos* obtêm-se os seguintes dados:

O valor para cada um está a seguir:

- Modo RX é a somatória de *Process* + RX + CN3063, pois o NSS para ativar o modo RX precisa dos demais citados, a um valor próximo a 23mA.
- Modo TX é *Process* + TX + CN3063, cujo a soma da um valor próximo a 31mA.
- Modo *Sleep* é o valor direto, sendo o consumo do *Sleep* do ATmega328 + CN3063 o valor de 120µA, quando conectado somente na bateria, e 800µA, quando carregando a bateria.

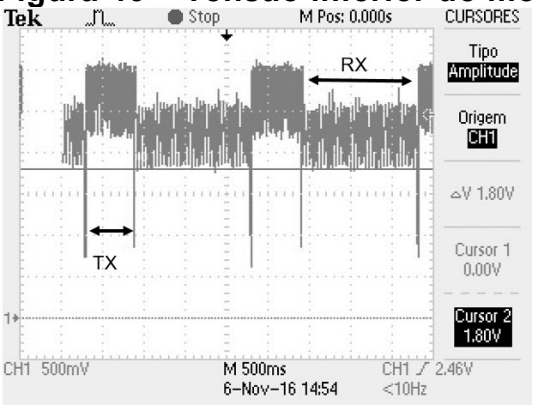
Verificando o consumo dos modos no osciloscópio obteve-se valores bem próximos ao valores coletados usando a configuração da Figura 32, estão apresentados nas figuras Figura 45, Figura 46 e Figura 47 os dados coletados nos três modos.

Figura 45 – Tensão superior do modo RX



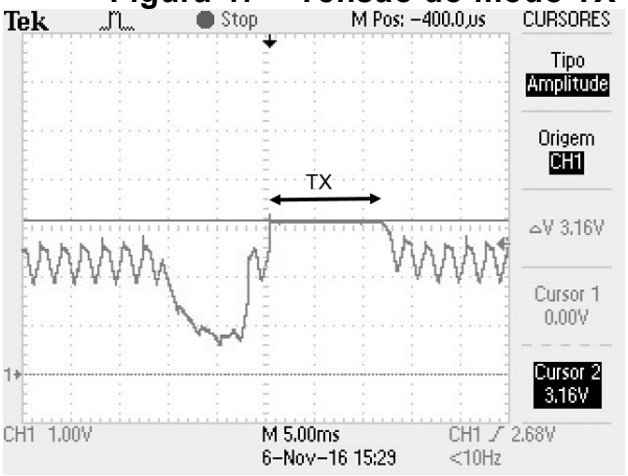
A Figura 45 mostra uma captura do osciloscópio, com o pico de consumo de RX. Como o shunt é 99 Ω, a corrente equivalente é de 25,8 mA.

Figura 46 – Tensão inferior do modo RX



Como na Figura 45 e Figura 46 o marcador do osciloscópio mostra a informação de RX. O valor do shunt continua 99Ω, com uma corrente equivalente a 18,1mA. Totalizando o consumo médio de RX, calculando a média dos dois valores obtidos através das Figura 45 e Figura 46, obtendo um valor médio de 21,95mA ((25,8+18,1) /2=21,95mA). Assim como foi usado para aferir o consumo do modo RX foi feito para TX, com peculiaridade de no modo TX, não possuir um delta na coleta, devido a sensibilidade da configuração usada, mostrada na Figura 47. E o modo TX com um consumo de 31,9mA (3,16V/99Ω), apresentado na Figura 47.

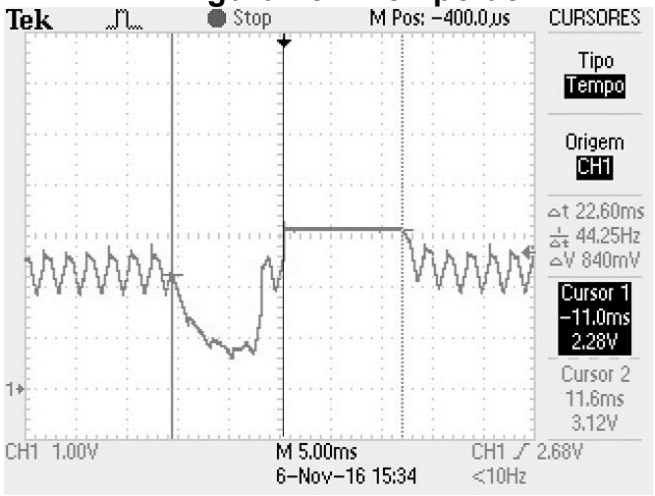
Figura 47 – Tensão do modo TX



Em relação aos tempos, esses variam de acordo com a aplicação. Somente o modo TX é constante, pois na plataforma Radiuno tem a transmissão de um pacote de dados fixo mais preâmbulo, no caso 64 Bytes. Uma informação importante é a taxa de

transmissão, pois ela é diretamente relacionada ao tempo de transmissão. Na maioria das aplicações não há mudança na taxa. A taxa de transmissão adotada é de 38000bps, com um tempo aproximado de 12ms (Figura 48). O delta entre RX e TX é devido ao tempo necessário para o desligamento do modo RX e o acionamento do TX, aproximadamente um tempo de 10ms.

Figura 48 – Tempo de TX

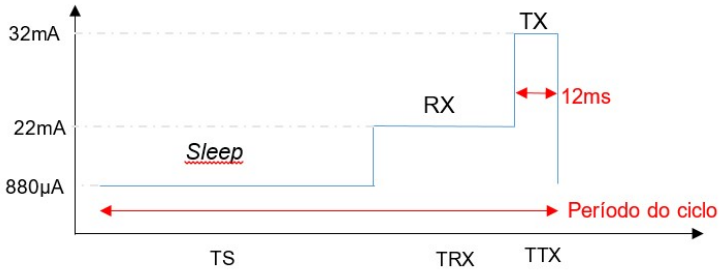


Resultados do consumo em cada modo:

- Modo TX um consumo de 32mA
- Modo RX um consumo de 22mA
- Modo *Sleep* um consumo de 880µA/120µA

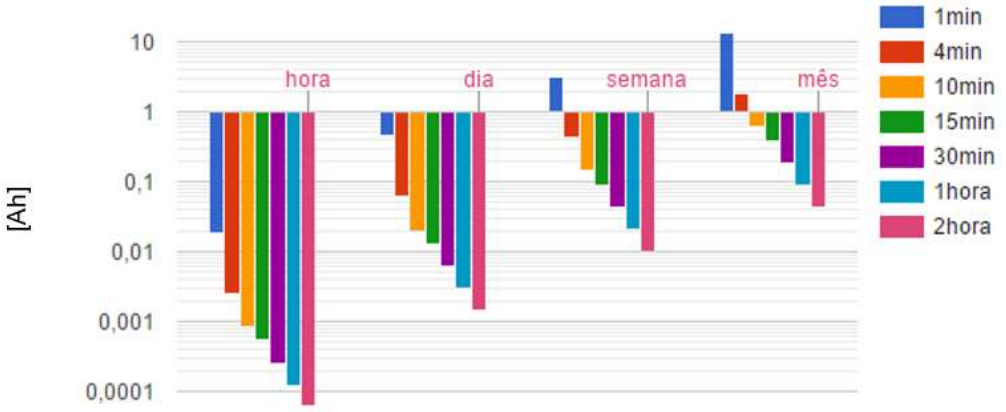
Para analisar o consumo de energia do ciclo de operação do nó sensor lembra-se que o período do ciclo pode ser ajustado conforme a aplicação (Figura 49).

Figura 49 – Consumo do nó em seus respectivos modos de operação



Usando as informações da Tabela 4 foi gerado um gráfico, no qual é possível ver as curvas de consumo conforme as taxas de amostragens, mostrado na Figura 50.

Figura 50 – Gráfico do consumo do NSS em escala logarítmica



No dimensionamento da bateria é possível estimar a capacidade da bateria necessária: uma aplicação com taxa de requisições a cada 10mim necessitando que o NSS suporte um mês sem recarga, é necessária uma bateria de 0,6Ah no mínimo.

Para o dimensionamento do sistema energético do NSS, é necessário estabelecer a taxa de requisições e o período sem carga da bateria, para encontrar o consumo do NSS. Para o dimensionamento foi definido uma coleta a cada 4mim e o período sem sol de 18 horas. O consumo do NSS encontrado fica conforme se segue:

- Primeiro com o uso da Equação 3.2 encontra a corrente média de 167μA que multiplicando pela tensão nominal de consumo (3,3V), obtém-se uma potência de 550μW;
- Usando a Equação 3.3 resulta no consumo energético de 1,98J, para as 18h (64800s) um valor de 128304J
- Com o consumo é realizado o dimensionamento da bateria usando a Equação 3.6, que resulta em uma bateria com capacidade mínima de 104mAH.
- O dimensionamento do PV resulta da energia necessária de 128304J e a produzida é obtida na Equação 3.4, usando a média diária de 4kW/m²/d considerando o Item 3.5, resultando em uma “colheita” de 360mW por célula ao dia

- Sendo a janela de sol disponível no local de 8 horas, então a energia coletada por célula é de 45mW (360mW/8h) por hora de sol. São necessários aproximadamente 384mW (104mAH*3.7) para suprir a demanda e no local só é possível coletar por 4 horas ao dia, cada célula vai produzir 45mW, em 4 horas 180mW,
- Portanto, sendo necessário 2 células somente para coletar a energia para a bateria.
- Sendo assim, para o funcionamento do NSS é no mínimo 3 células 2 para a bateria e uma para alimentar o NSS enquanto as 2 estiverem carregando a bateria, sendo que o consumo é de 550µW do NSS. No PV foi utilizado 4 células para dar uma margem maior de segurança.

O consumo do NSS (Tabela 6), usado para determinar a energia necessária para o dimensionamento do PV e da bateria do NSS, mostra que o consumo do NSS usando intervalos menores tendem ao consumo do modo RX e medidas com intervalos maiores tendem ao consumo de *sleep*.

Tabela 6 – Consumo dos modos do NSS e do Controlador de carga

Modo de operação	Corrente em Amperes
CN3063 V bat	120µ
CN3063 V in	800µ
Process	9m
Sleep	130µ
RX	13m
TX	22m

Na Tabela 6 os dois primeiros valores são relativos ao controlador de carga, dos quais sempre estão consumindo a energia. Observar-se que, o controlador tem um menor consumo quando está conectado a uma bateria.

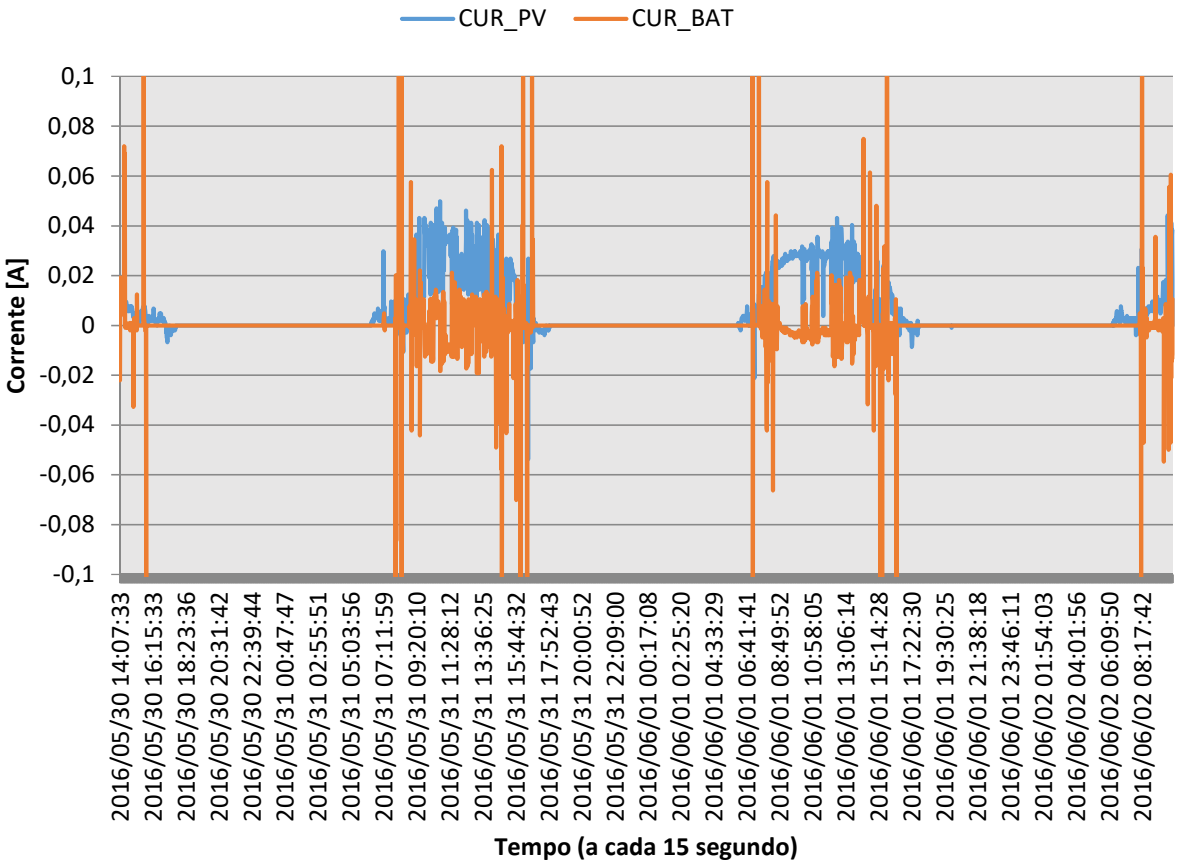
Foram utilizadas 4 configurações na bancada que estão apresentadas nas tabelas a seguir, seguidas dos gráficos de fluxo de energia obtidos através dos testes. A Tabela 7 mostra a primeira configuração usada.

Tabela 7 – Configuração 1 da bancada

Quantidade de células fotovoltaicas	1
Tempo de amostragem	15 segs.
Intervalo de transmissão	20 segs.
Carga	BE900
Direção do PV	Norte
Inclinação do PV	30 graus
Sleep	Não
Quantidade de células de bateria	1

Esta foi a primeira configuração, exigente frente ao número de painéis (Figura 51). Observar que uma única célula não é suficiente para carregar a bateria e, dependendo do período, nem suportando a carga, como no dia 31. As correntes negativas da bateria representam a energia sendo armazenada e, quando positivas representam o fornecimento da energia à carga. Quanto a corrente do PV, representa a energia coletada, não possuindo valores negativos.

Figura 51 – Gráfico de fluxo de energia conforme configuração 1

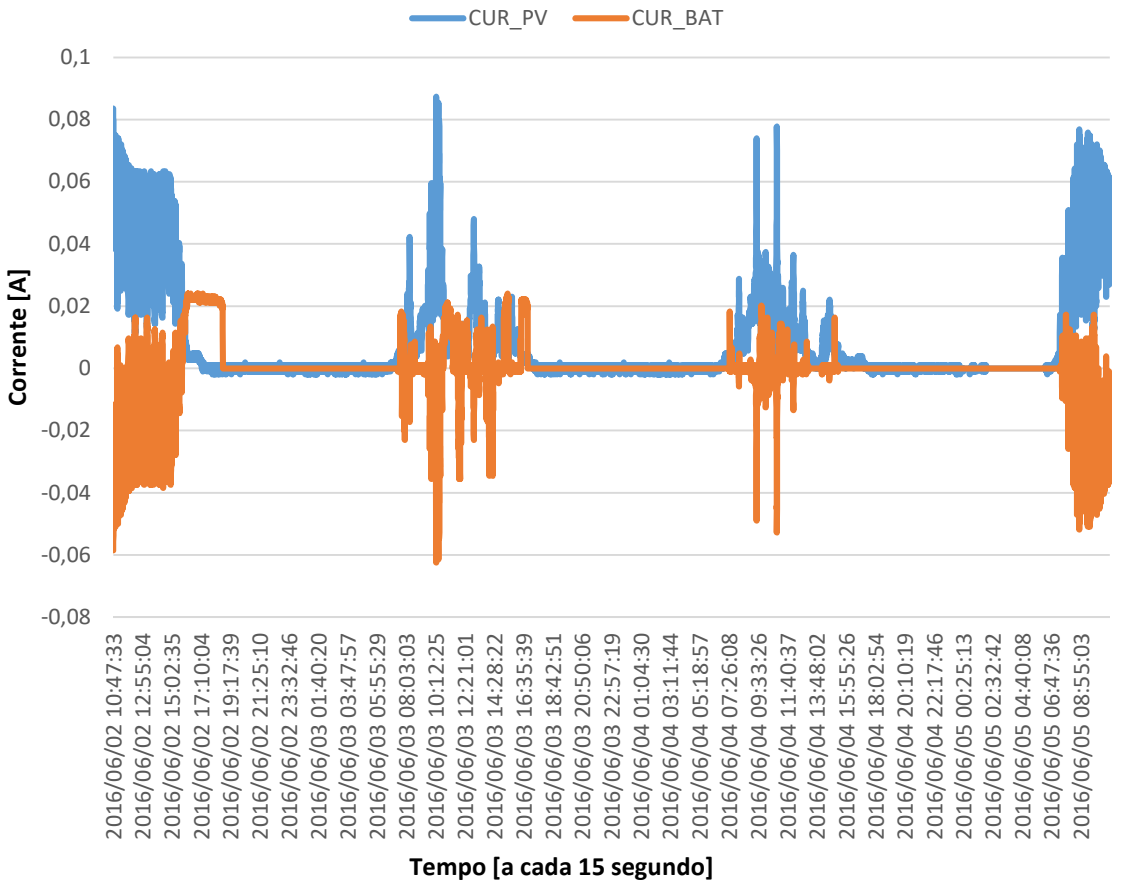


O segundo teste teve a configuração conforme a Tabela 8. Acrescentou-se uma célula e o fluxo foi alterado (Figura 52). Ainda com duas células não foi o suficiente para começar a carregar a bateria, porém é possível observar que a bateria começa a apresentar uma pequena sobrevida após a ausência da energia provida do PV

Tabela 8 – Configuração 2 da bancada

Quantidade de células fotovoltaicas	2
Tempo de amostragem	15 segs.
Intervalo de transmissão	20 segs.
Carga	BE900
Direção do PV	Norte
Inclinação do PV	30 graus
Sleep	Não
Quantidade de células de bateria	1

Figura 52 – Gráfico de fluxo de energia conforme configuração 2

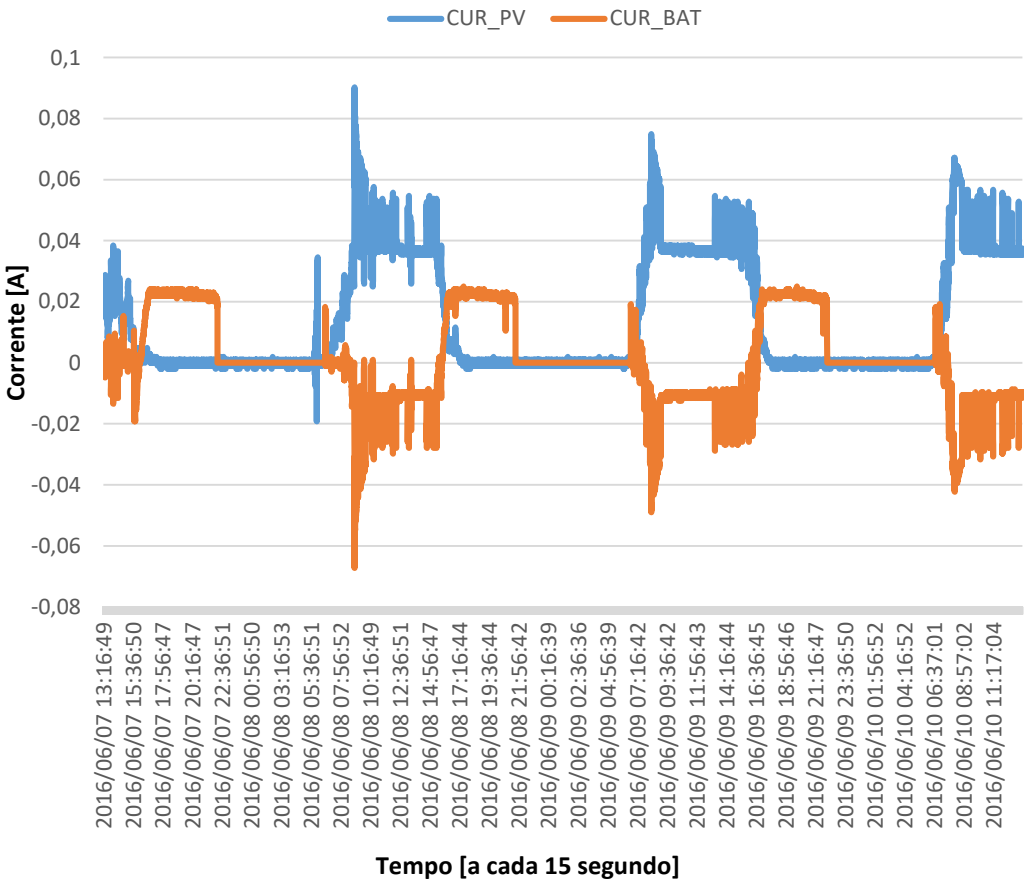


O terceiro teste teve a configuração conforme a Tabela 9. Adicionando mais uma célula, o fluxo, diferente dos testes anteriores, já começa a funcionar a dinâmica de carga da bateria para a sobrevida da carga quando a energia do PV cessa (Figura 53).

Tabela 9 – Configuração 3 da bancada

Quantidade de células fotovoltaicas	3
Tempo de amostragem	15 segs.
Intervalo de transmissão	20 segs.
Carga	BE900
Direção do PV	Norte
Inclinação do PV	30 graus
Sleep	Não
Quantidade de células de bateria	1

Figura 53 – Gráfico de fluxo de energia conforme configuração 3



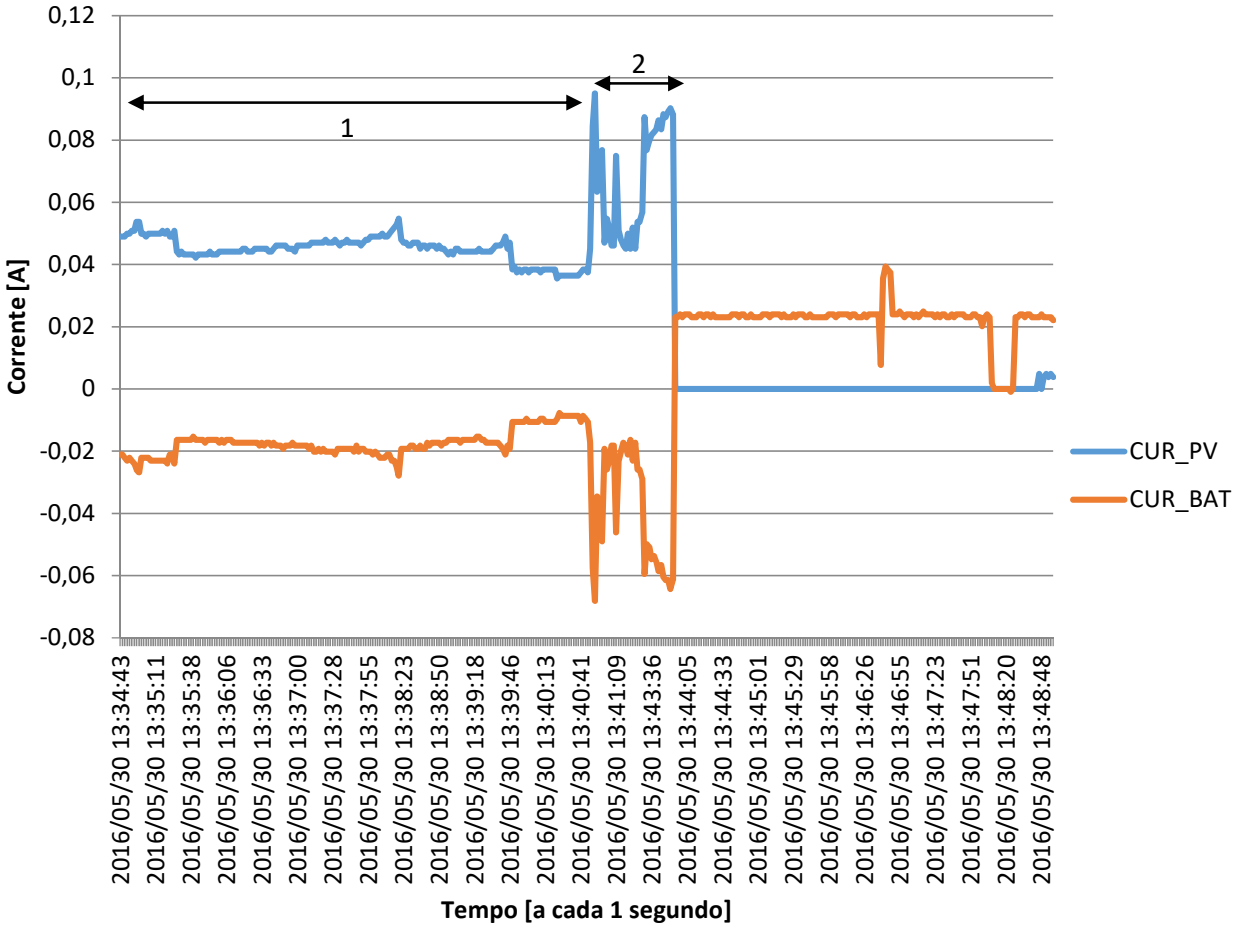
E o último teste foi considerado conforme Tabela 10. O PV foi substituído por uma fonte de tensão. A Figura 54 mostra a dinâmica para avaliar a corrente máxima que o

sistema coleta. A bateria começa a carregar com uma corrente menor (período 1) e depois segue para o máximo que o sistema suporta (período 2).

Tabela 10 – Configuração 4 da bancada

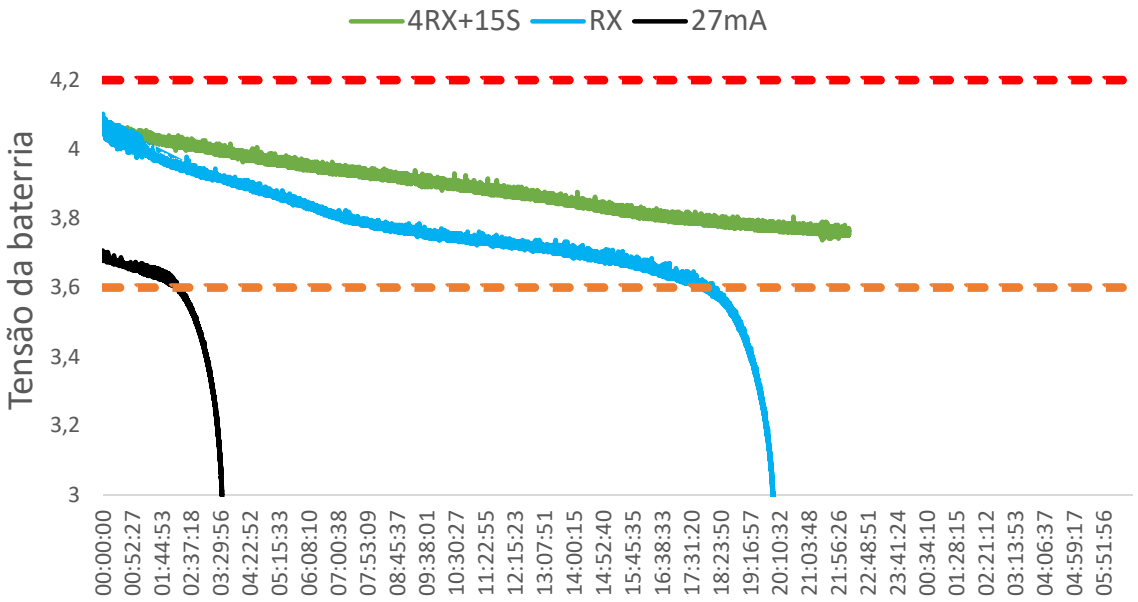
Quantidade de células fotovoltaicas	---Fonte Controlada---
Tempo de amostragem	1 segs.
Intervalo de transmissão	20 segs.
Carga	BE900
Direção do PV	----
Inclinação do PV	----
Sleep	Não
Quantidade de células de bateria	1

Figura 54 – Gráfico de fluxo de energia conforme configuração 4



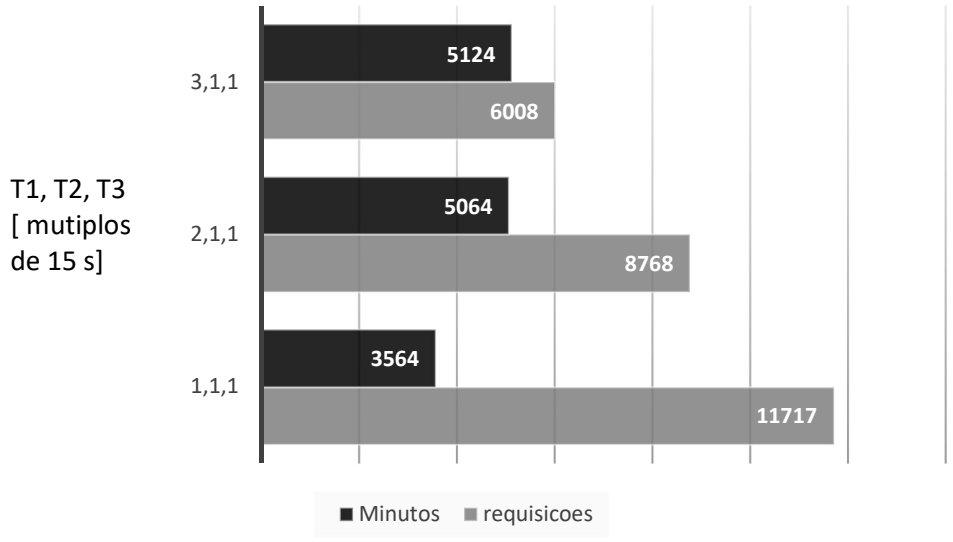
A seguir (Figura 55) está o resultado de descarga da bateria.

Figura 55 – Testes de descarga de bateria sem *sleep2*



No teste de descarga da bateria sem a implementação do sleep2 (Figura 55) constata-se que para um consumo de uma carga resistiva de 27 mA, a bateria de 500 mAH dura apenas 3 h 30 mim. Usando o módulo BE900 no modo RX, modo normalmente usado quando o rádio não tem implementado o *sleep*, a bateria tem uma vida de 20 h. O último teste foi usado com uma alternância dos estados para verificar a contribuição do *sleep* na economia, que superou um dia de vida. Foi realizado mais um teste de descarga da bateria (Figura 56), porém agora já no NSS e com o sleep2 implementado.

Figura 56 – Teste de descarga da bateria usando *sleep*2



Com o sleep2 em sua configuração, que consome mais a vida da bateria dura 3564 m equivalente a aproximadamente 60 horas, mais que 2 dias. E conforme se alteram as configurações, mais tempo de vida a bateria suporta. Porém, para uma maior vida do NSS, menor é a quantidade de requisições feitas.

- Antena

O parâmetro S11 da antena do NSS mede a perda de retorno por volta de 20dB, que é um valor bom (100 vezes a potência útil em relação à potência refletida). A frequência que obteve essa perda de retorno está próxima de 915 MHz, que é a frequência central. A imagem do *vector network analyzer* utilizado (Figura 57) mostra que o vale da amostragem da perda de retorno está em 916,27 MHz com o valor de 21,97 dB, porém é possível notar que em 915 MHz, que é o centro da banda, a perda de retorno é próxima de 20 dB. O tipo desta antena é setorial (Figura 58).

Figura 57 – Parâmetro S11 da antena do NSS

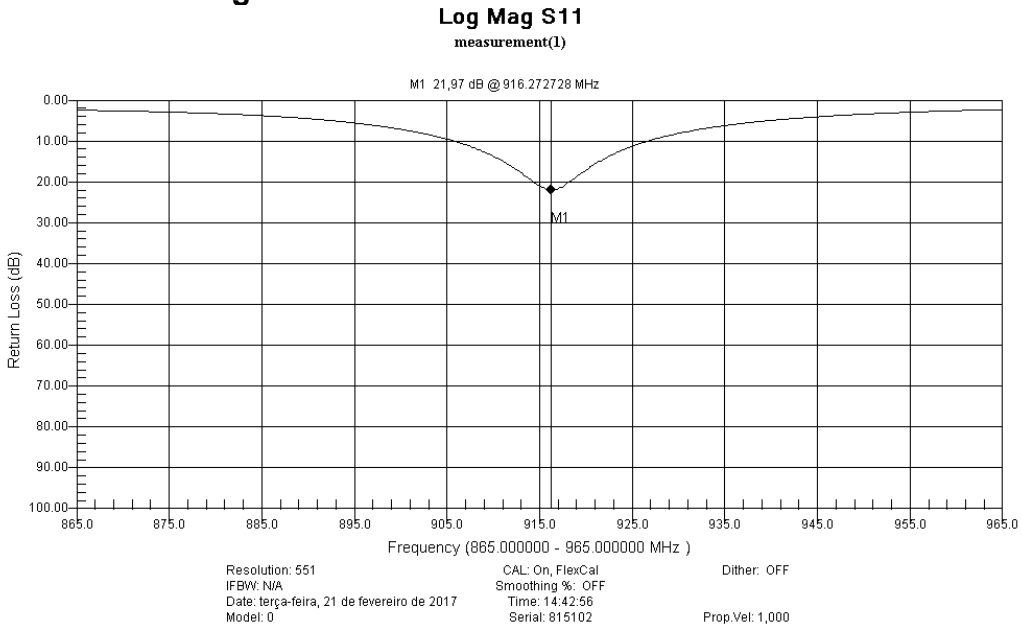
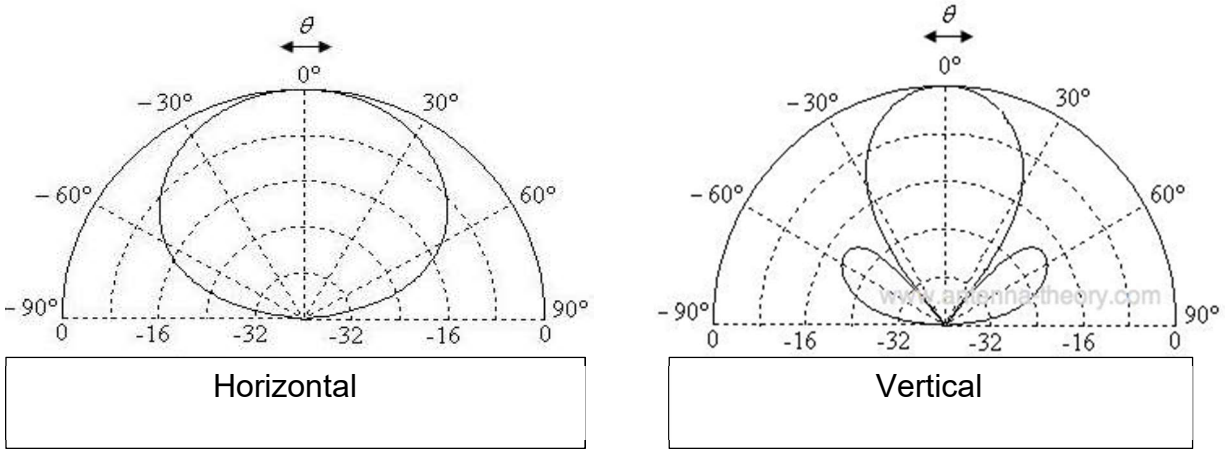


Figura 58 – Diagrama de radiação da antena



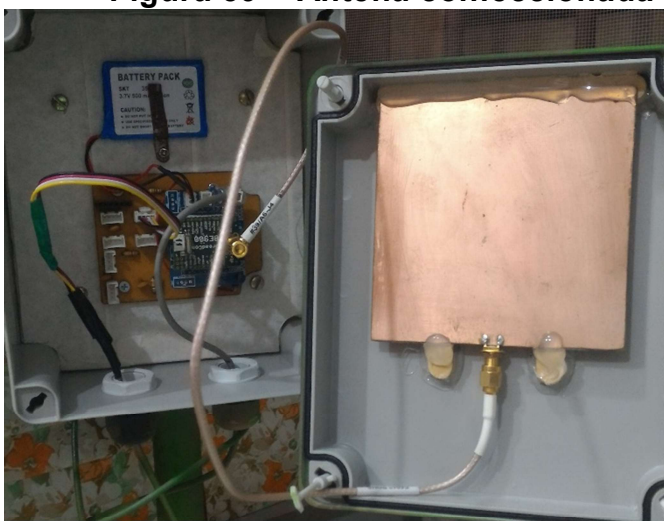
Fonte: “Microstrip Antennas: The Patch Antenna”, [s.d.]

Com o diagrama horizontal e vertical na Figura 58 é possível saber a característica setorial da antena, com essa informação é possível fazer um apontamento mais eficiente. Em alguns casos a antena já vem com uma estrutura mecânica para esse apontamento.

Para avaliar o ganho da antena foi usada uma antena referência de 12,5dBi, que resultou uma RSSI média de -25dBm quando a antena confeccionada uma média de

-35dBm. Com o resultado a antena confecciona apresenta um ganho de 2,5dBi. Embora o ganho não seja muito alto, a vantagem é que a mesma pode ser montada dentro da caixa, evitando problemas com infiltração de agua, como pode acontecer com sensores em ambiente externo. A Figura 59 mostra a antena dentro do NSS.

Figura 59 – Antena confeccionada

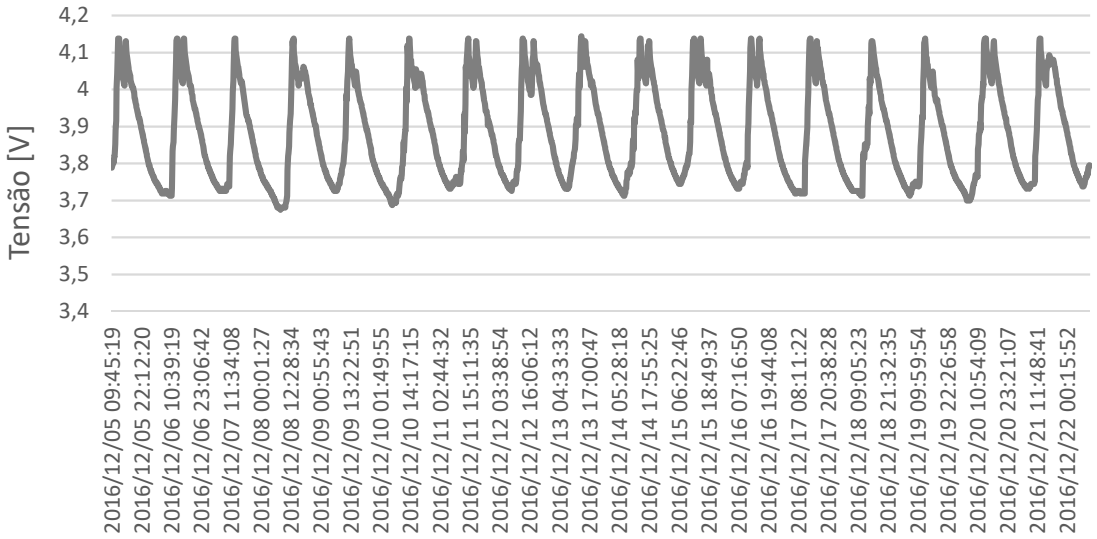


Observar que na Figura 59 é possível avaliar que mecanicamente a antena ficou bem acomodado dentro da caixa do NSS.

5.2 PONTO DE COLETA 1

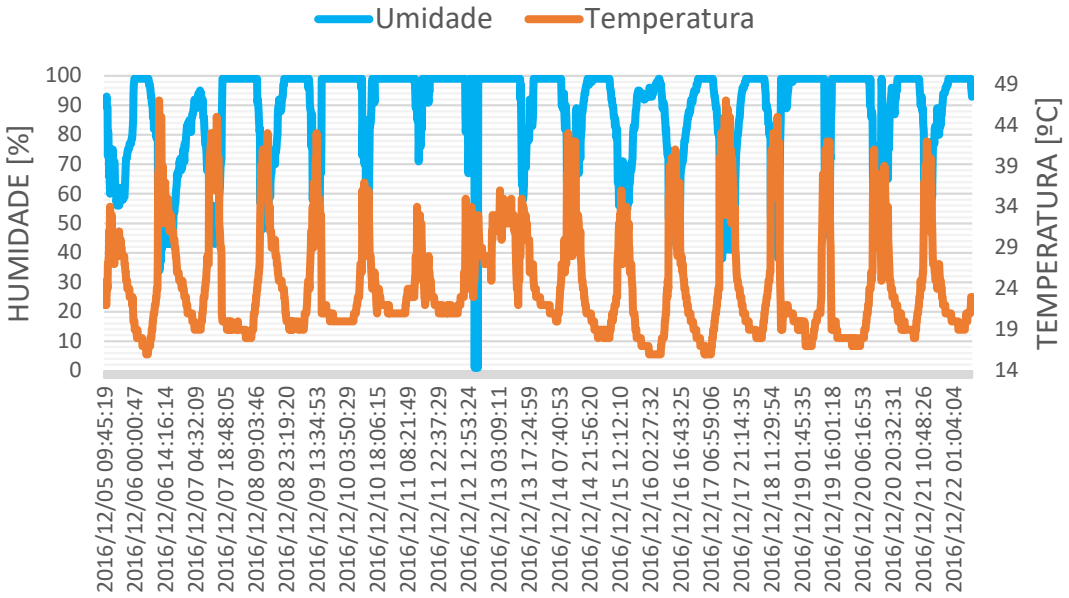
Os dados coletados de 2016/12/05 09:45:19 à 2016/12/22 09:42:30, conforme o Item 4.3.2, tem dados referente a tensão da bateria, umidade e temperatura. Começando com a tensão da bateria (Figura 60). A tensão da bateria teve um comportamento cíclico, no qual o nível de tensão da bateria variou diariamente, com a janela de carga média de 6 horas, no período entre 8 h e 14 h. Na coleta o valor máximo de tempo entre uma medida e outra é de 0h:4m:12s, não havendo intervalos maiores, como o configurado era de uma taxa de requisição a cada 4min, não houve interrupção do sistema.

Figura 60 – Tensão da bateria na primeira coleta no ponto 1



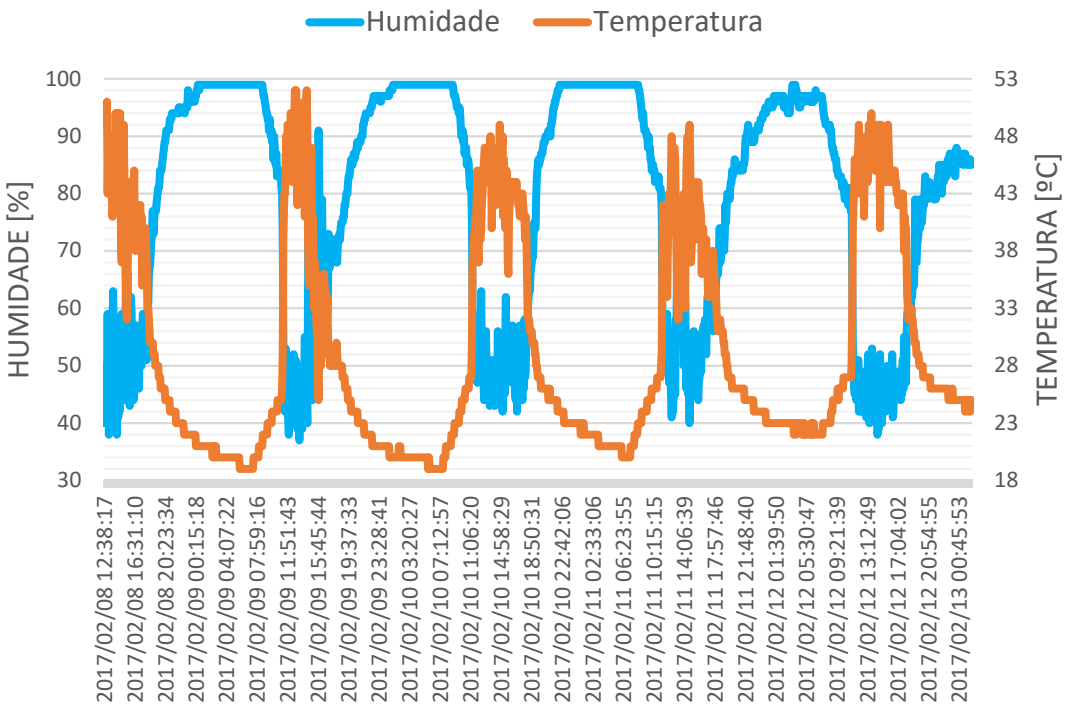
A seguir a Figura 61 mostra a temperatura e umidade. Com o gráfico abaixo é possível constatar a relação da umidade com a temperatura conforme esperado, com ciclos variando de 47 à 16 graus e a umidade por volta de 30 até 100%. Não sendo objeto desta dissertação a análise deste resultado. É possível observar que houve um *outlier* no dia 12/12, que é um valor que deve ser desprezado.

Figura 61 – Temperatura e umidade primeira coleta no ponto 1



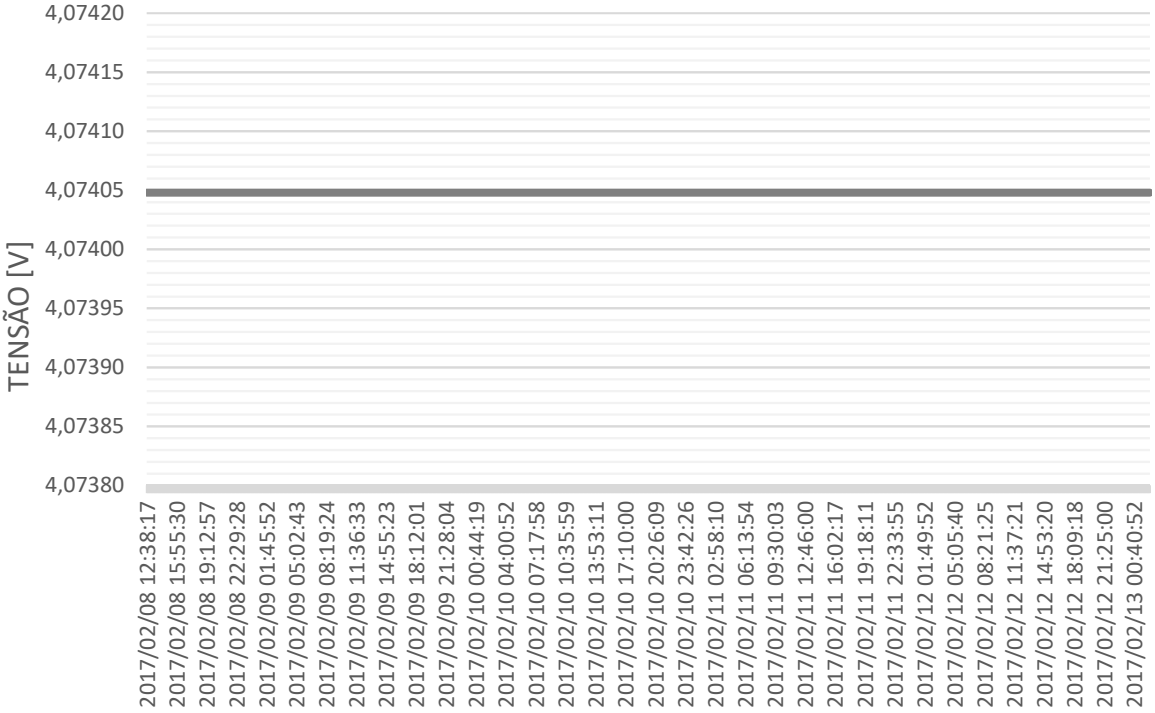
Foi realizado mais quatro dias de coleta de 2017/02/08 12:38:17 à 2017/02/13 02:21:10 apresentados a seguir, porém agora com uma amostragem a cada 5mim e implementado o *Sleep2* como mostra a Figura 62 com a temperatura e umidade.

Figura 62 – Temperatura e umidade segunda coleta no ponto 1



Os valores se mantiveram consistentes como nos das medidas anteriores. O valor máximo de tempo entre uma medida e outra foi de 0h:5m:34s, não havendo intervalos maiores, não houve interrupção do sistema. Houve pouca perda de pacote, que não comprometeu as coletas, sendo normal a perda de pacotes em RSSF. A Figura 63 mostra a tensão da bateria.

Figura 63 – Tensão da bateria segunda coleta no ponto 1

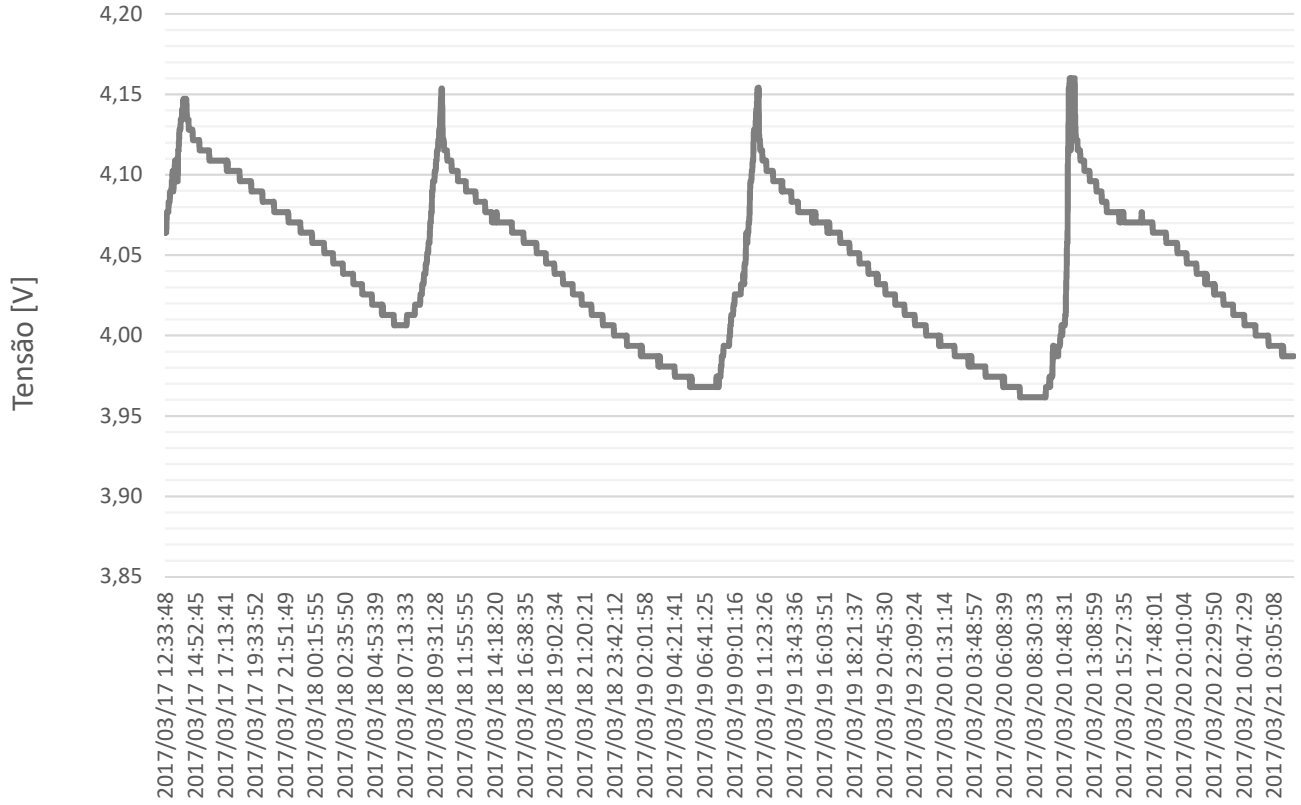


Pode se observar a constância na tensão da bateria, representando que a mesma se manteve carregada o tempo todo como demonstra a Figura 63. O consumo da carga foi muito menor se comparada à primeira coleta.

5.3 PONTO DE COLETA 2

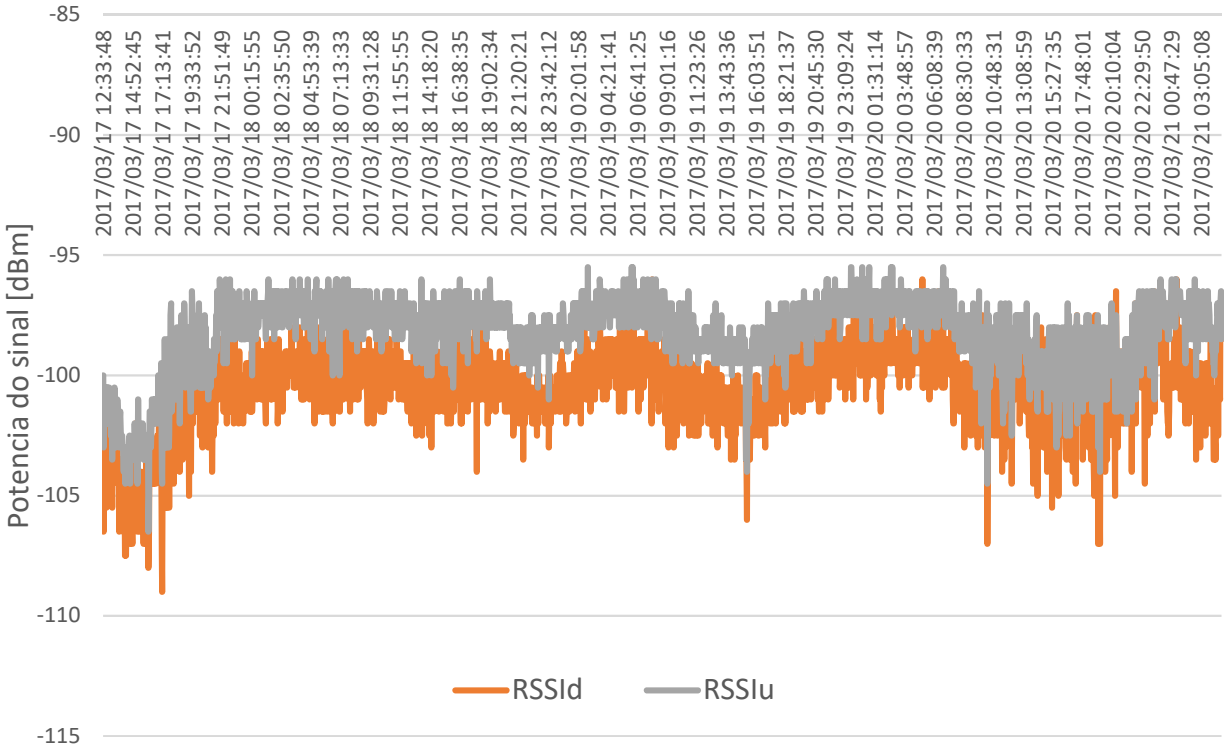
No ponto 2 de coleta foram coletadas as mesmas grandezas, com o adicional da informação da RSSI do link. A primeira grandeza coletada é a da bateria como mostra o gráfico da Figura 64.

Figura 64 – Tensão da bateria no ponto 2



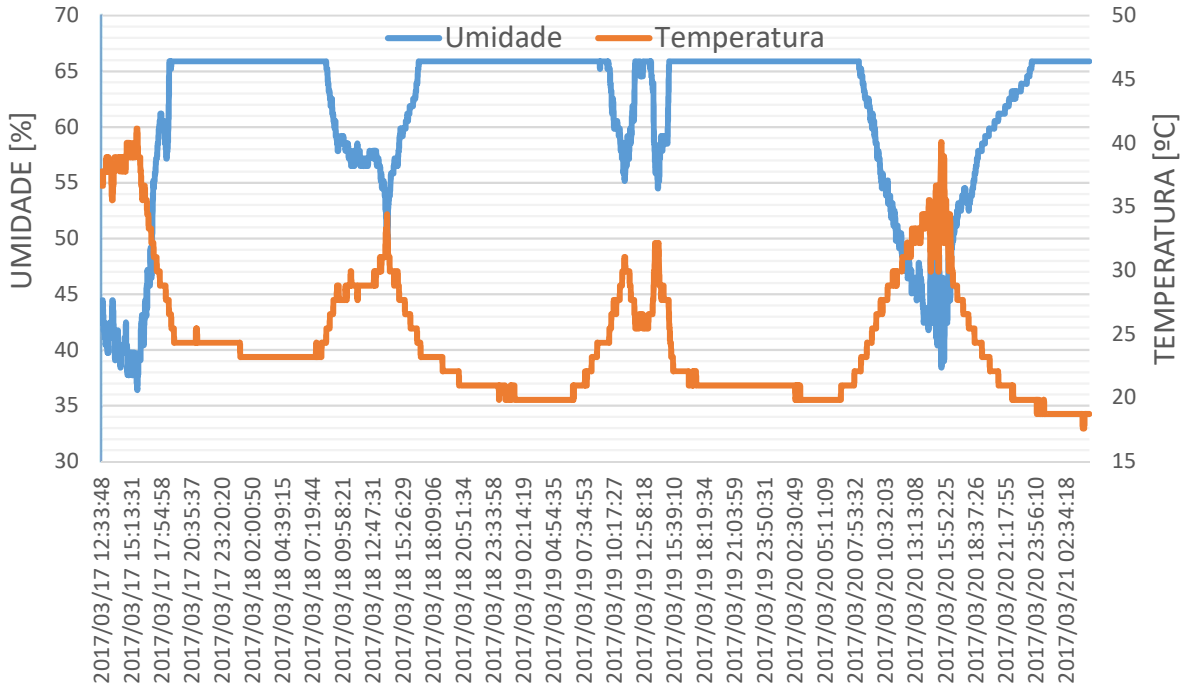
A janela de carga é de 4 horas entre o período das 8 às 12 horas, diferente da coleta 1 que era 6 horas, isso se deve ao fato de que o local da instalação do NSS não estar em condições ideais. No ponto 2 foi armazenado os valores da RSSI do link radio como mostra o gráfico na Figura 65.

Figura 65 – RSSI no ponto 2



Uma melhora na RSSI pode ser observada no período noturno. Os valores entre -95 e -105 dBm representa uma conexão ruim, mas o link atendeu seu propósito. Outra informação coletada é a temperatura e umidade como mostra a Figura 66.

Figura 66 – Temperatura e umidade do ar no ponto 2



Assim como as demais temperatura e umidade coletadas. Nesta coleta em específico o transdutor saturou em 67% a umidade, não coletando medidas superiores. Esse é um erro que deve ser avaliado quando for utilizado esse tipo de sensor. Lembrando que não faz parte desta dissertação uma análise mais profunda da temperatura e umidade dos pontos de coleta.

6 CONCLUSÃO

Como pode ser constado pelos resultados obtidos, foi possível a construção de um NSS, que atendeu ao esperado pela proposta do trabalho. As estratégias de *sleep* avaliadas demonstram que o *sleep2* foi superior ao *sleep1*. É viável a construção de um nó sensor sem fio alimentado por bateria e carregado por painel fotovoltaico. O uso de RSSF facilita as coletas de dados; e em caso de futura necessidade de monitorar-se mais grandezas, é necessário apenas que se adicionem os transdutores necessários no NS/NSS.

6.1 AVALIAÇÃO DA CONCEPÇÃO SISTÊMICA

A concepção sistêmica se mostrou adequada para a coleta de dados no meio ambiente. O sistema desenvolvido, foi avaliado em dois pontos de coletas, observando suas peculiaridades, como distâncias do link rádio e local de instalação. Com uma metodologia que se repetiu nos dois pontos de coleta, demostrando a eficiência do artefato desenvolvido. O sistema usado se manteve ativo, não sofrendo nenhum tipo de descontinuidade nos dados coletados.

Como um elemento autônomo do ponto de vista energético, o NSS supriu essa necessidade.

6.2 TRABALHOS FUTUROS

A rede usada só possuía um NSS. Um trabalho futuro é o desenvolvimento de uma rede com mais NSS.

O alcance da rede neste trabalho estava limitado a apenas o raio de alcance da base. O trabalho pode ser estendido com a introdução da comunicação por saltos, podendo ser implementados *clusters heads* para aumentar o alcance rádio da rede.

Outro fator que pode ser melhorado é a interface homem máquina, fazendo com que o usuário visualize os dados de maneira mais adequada, como por exemplo, a inclusão de um *dashboard* para a exibição dos dados. Com esta evolução é possível a criação de alarmes para situações críticas.

7 REFERÊNCIA

ABBAS, M. M. et al. Solar Energy Harvesting and Management in Wireless Sensor Networks. **International Journal of Distributed Sensor Networks**, v. 2014, 2014.

ABBEY, C.; JOOS, G. Supercapacitor energy storage for wind energy applications. **IEEE Transactions on Industry Applications**, v. 43, n. 3, p. 769–776, 2007.

AKYILDIZ, I. F. et al. A Survey on Sensor Networks. **IEEE Communications Magazine**, v. A CCEPTED, n. 2, p. 102–114, 2002.

ANRITSU. VNA Master ™ MS2024A/MS2026A and MS2034A/MS2036A Handheld Vector Network and Spectrum Analysis for General Purpose Applications 2 Feature Models Benefit. [s.d.].

ATMEL. **ATMEGA328 description, ATMEGA328 datasheets**. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/392243/ATMEL/ATMEGA328.html>. Acesso em: 5 maio. 2017.

AZEVEDO, G. M. S. et al. Evaluation of maximum power point tracking methods for grid connected photovoltaic systems. **Power Electronics Specialists Conference, 2008. PESC 2008. IEEE**, p. 1456–1462, 2008.

BACCELLI, F.; BLASZCZYSZYN, B.; MÜHLETHALER, P. An Aloha protocol for multihop mobile wireless networks. **IEEE Transactions on Information Theory**, v. 52, n. 2, p. 421–436, 2006.

BOCCHI, N.; FERRACIN, L. C.; BIAGGIO, S. R. Pilhas e Baterias: Funcionamento e Impacto Ambiental. **Química Nova na Escola**, v. 11, p. 3–9, 2000.

BRITO, M. DE; LUIGI, G. Avaliação das principais técnicas para obtenção de MPPT de painéis fotovoltaicos. **9th IEEE/IAS International Conference on Industry Applications - INDUSCON 2010 -**, p. 6, 2010.

CONSONANCE. **Lithium Ion Battery Charger for Solar-Powered Systems CN3063 General Description**. [s.l: s.n.]. Disponível em: <http://www.consonance-elec.com/pdf/datasheet/DSE-CN3063.pdf>. Acesso em: 7 mar. 2017.

Corrente de Pilhas Comuns (DUV373). Disponível em: <http://www.newtoncbraga.com.br/index.php/duvidas-dos-internautas/5723-corrente-de-pilhas-comuns-duv373.html>. Acesso em: 24 abr. 2017.

DELGADO GOMES, R. et al. Correlation between spectral occupancy and packet error rate in IEEE 802.15.4-based industrial wireless sensor networks. **IEEE Latin America Transactions**, v. 10, n. 1, p. 1312–1318, 2012.

DUFFIE, J. A.; BECKMAN, W. A.; WOREK, W. M. **Solar Engineering of Thermal Processes, 4nd ed.** [s.l: s.n.]. v. 116

ELIANE, P.; FARIA, A.; FADIGAS, A. **PEA –2420 PRODUÇÃO DE ENERGIA Energia Solar Fotovoltaica : Fundamentos, Conversão e Viabilidade técnico-econômica**. 2004Disponível em: <https://edisciplinas.usp.br/pluginfile.php/56337/mod_resource/content/2/Apostila_solar.pdf>. Acesso em: 17 abr. 2017

GROSSMANN, C. **Maior usina solar do mundo é inaugurada nos Emirados Árabes Unidos | HypeScience**. Disponível em: <http://hypescience.com/maior-usina-solar-do-mundo/>. Acesso em: 7 dez. 2016.

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645–1660, set. 2013.

Home - Radiuino. Disponível em: <http://radiuino.cc/>. Acesso em: 26 maio. 2017.

JACOBI, P. Educação ambiental, cidadania e sustentabilidade. **Caderno de Pesquisa,** v. 118, p. 189–205, 2003.

Link Budget Calculator. Disponível em: <https://www.pasternack.com/t-calculator-link-budget.aspx>. Acesso em: 18 jul. 2017.

LIU, T. Digital-output relative humidity & temperature sensor/module Capacitive-type humidity and temperature module/sensor. [s.d.].

LUCAS, J.; SILVA, D. ESTUDO DO COMPORTAMENTO DE ANTENA DE MICROFITA COM SUBSTRATO METAMATERIAL. 2015.

MACCHIARELLI, M. Solar Powered Wireless Sensors & Instrumentation : Energy Harvesting Technology Reduces Operating Cost at Remote Sites. [s.d.].

MAINWARING, A. et al. Wireless Sensor Networks for Habitat Monitoring. [s.d.].

MARIANA LETTI. **Seminário 2 - Uso de tecnologias: as redes**. Disponível em: <http://alpoadm.wixsite.com/diariodebordo/single-post/2013/09/10/Seminário-2-Mariana-Letti>. Acesso em: 18 jul. 2017.

MARJOYA. **Relojes con sistema de energía solar**. Disponível em: <http://www.marjoya.com/blog/2016/01/25/relojes-sistema-energia-solar/>. Acesso em: 18 jul. 2017.

MEDEIROS, A.; MEDEIROS, C. F. DE. Possibilidades e Limitações das Simulações Computacionais no Ensino da Física. **Revista Brasileira de Ensino de Física**, v. 2, 2002.

Microstrip Antennas: The Patch Antenna. Disponível em: <http://www.antenna-theory.com/antennas/patches/antenna.php>. Acesso em: 20 fev. 2017.

Microstrip Patch Antenna Calculator. Disponível em: <https://www.pasternack.com/t-calculator-microstrip-ant.aspx>. Acesso em: 20 abr. 2017.

MIGUEL, J.; GOMES, S. Power-Film: Um filme flexível autónomo para alimentar dispositivos elétricos Universidade do Minho Escola de Engenharia. 2012.

MORAES, D. M. DE; TARRACHELLI, P. E. DE A. **Reprodução Computadorizada dos Movimentos da Mão - Accelerometer - YouTube**. Disponível em: <https://www.youtube.com/watch?v=MZJ4tc5VqC4>. Acesso em: 3 mar. 2017.

MORALES, D. S. **Maximum Power Point Tracking Algorithms for Photovoltaic Applications**. [s.l.] aalto university, 2010.

NALLUSAMY, R.; DURAISWAMY, K. Solar Powered Wireless Sensor Networks for Environmental Applications with Energy Efficient Routing Concepts: A Review. **information technology journal**, v. 10, 2011.

NASCIMENTO, C. A. DO. Princípio De Funcionamento Da Célula. 2004.

PINHO, J. T.; GALDINO, M. A. Manual de Engenharia para Sistemas Fotovoltaico. 2014.

PINTO, C. et al. **Resumo Palavras-Chave**. [s.l.] Faculdade de Engenharia da Universidade do Porto, 2015.

RADIOIT. **datasheet - BE900**. Campinas: [s.n.]. Disponível em: <http://radioit.com.br/wp-content/uploads/2015/06/BE900-Datasheet-v1.5.pdf>. Acesso em: 8 jun. 2016.

RAGHUNATHAN, V.; KANSAL, A.; HSU, J. Design considerations for solar energy harvesting wireless embedded systems. **Proceedings of the 4th …**, v. 0, n. C, p. 457–462, 2005.

RF MONOLITHICS. Sizing solar energy harvesters for wireless sensor networks. **Http://Www.Rfm.Com/Products/Apnotes/Anm1002.Pdf**, 2010.

RIBEIRO, JOSÉ E. **Universidade Federal de São Carlos Universidade Federal de São Carlos Middleware de Serviços Multi-Camadas para Redes de Sensores Sem Fio**. Sã Carlos: [s.n.].

RUIZ, L. B. et al. Arquiteturas para Redes de Sensores Sem Fio. 2004.

SALAMONI, I. T. METODOLOGIA PARA CÁLCULO DE GERAÇÃO FOTOVOLTAICA EM ÁREAS URBANAS APLICADA A FLORIANÓPOLIS E BELO HORIZONTE. 2004.

SEEED. **0.5W Solar Panel 55x70 - Power Supply - Seeed Studio**. Disponível em: <https://www.seeedstudio.com/0.5W-Solar-Panel-55x70-p-632.html>. Acesso em: 4 maio. 2017a.

SEEED. **Wireless Sensor Node - Solar Kit - Seeed Wiki**. Disponível em: <http://wiki.seeed.cc/Wireless_Sensor_Node-Solar_Kit/>. Acesso em: 7 mar. 2017b.

SOUSA, M. P.; LOPES, W. T. A. L. Desafios em redes de sensor sem fio. **REVISTA DE TECNOLOGIA DA INFORMAÇAO E COMUNICAÇAO**, v. 1, 2011.

STELLBOGEN, D. Use of Pv Circuit Simulation for Fault Detection in Pv Array Fields. **IEEE Photovoltaic Specialists Conference**, p. 1302–1307, 1993.

WIN, K. K. W. K. K. et al. Efficient solar energy harvester for wireless sensor nodes. **Communication Systems ICCS 2010 IEEE International Conference on**, p. 289–294, 2010.

8 APÊNDICE

8.1 APÊNDICE1 – CÓDIGO EM PYTHON

```
1
2   # coding: utf-8
3
4   # # Software de aquisição de dados#
5   # Este software tem como objetivo se conectar a um no sersor e salvar em banco de
    dados as informações coletadas
6   #
7   # ## Bibliotecas necessarias ##
8
9   # In[1]:
10
11  # %load Teamplate Pooling Simples Py 3_5.py
12  #!/usr/bin/env python
13
14  ################################## Bibliotecas
    ############################################################
15  import serial
16  import datetime
17  import sqlite3
18  import math
19  import time
20  import struct
21  import os.path
22  #from ipywidgets import interact
23
24
25  # ## Funções
26
27  # In[2]:
28
29  ################################## FUNCAO PARA VEREFICAR PORTAS ATIVAS
    ###################################
30  def verifica_portas():
31
32      portas_ativas = []
33      for numero in range(100):
34
35          try:
36              objeto_verifica = serial.Serial(numero)
37              portas_ativas.append((numero, objeto_verifica.portstr))
38              objeto_verifica.close()
39
40          except serial.SerialException:
41              pass
42      return portas_ativas
43
44
45  # In[3]:
46
47  ################################## FUNCAO PARA EXECUTAR POOLING
    ###################################
48  def realizar_medidas(medidas,qtd,sensor,mac0,mac1,mac2,mac3):
49      controle = {}
50      Pacote = {}
51      for i in range(0,52): # faz um array com 52 bytes
52          Pacote[i] = 0
53          controle[i] = 0
54      for j in range(0,medidas):
55          for k in range(0,qtd):
56              Pacote[8] = sensor+k # ID Sensor, Byte[8]
57              Pacote[10] = 0 # ID de quem transmite, Byte[10]
58
59  ######################################### Sleep configuraçao do sensor
    ###################################
60              Pacote[4] = mac0 # 1 para ativar o sleep
61              Pacote[5] = mac1 # Sleep2 = T1
62              Pacote[6] = mac2-1 # Sleep2 = T2  # tempo do sleep
63              Pacote[7] = mac3 # Sleep2 = T3
64
65
66  ############################ TRANSMISSÃO DO PACOTE
    ##########################################
```

```
67              ser.flushInput() # Limpa o buffer da serial
68              for z in range(0,52):
69                  TXbyte = bytes([Pacote[z]])
70                  ser.write(TXbyte)
71
72  ############################## RECEPCAO DO PACOTE
    ##############################################
73              time.sleep(0.5) # Aguarda resposta do sensor (tempo de time out em
                segundos)
74              line = ser.read(52) # faz a leitura de 52 bytes do buffer que recebe da
                serial pela COM
75              if len(line) == 52: # Checa se recebeu 52 byte
76                  Anser = str(line[10]) # ID de quem transmite, Byte[10]
77                  c.execute("CREATE TABLE IF NOT EXISTS Sensor"+Anser+"(Data TEXT,
                    Tipo INTEGER, Valor INTEGER, IDCS INTEGER)")
78                  rssid1 = int(line[0]) # RSSI_DownLink, Byte[0]
79                  rssiu1 = int(line[2]) # RSSI_UpLink, Byte[2]
80                  if rssid1 > 128:                    #RSSI DownLink
81                      RSSId1=((rssid1-256)/2.0)-74     #RSSI DownLink
82                  else:                               #RSSI DownLink
83                      RSSId1=(rssid1/2.0)-74           #RSSI DownLink
84                  if rssiu1 > 128:                    #RSSI Uplink
85                      RSSIu1=((rssiu1-256)/2.0)-74     #RSSI Uplink
86                  else:                               #RSSI Uplink
87                      RSSIu1=(rssiu1/2.0)-74           #RSSI Uplink
88                  print ('Sensor',Anser,':',time.asctime(),':',' RssiU = ',RSSIu1,
                    'dBm ',' RssiD = ',RSSId1, 'dBm ') # Apresentaçcao no shell
89                  c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor) VALUES (?,
                    ?, ?)",(time.strftime("%Y/%m/%d %H:%M:%S"),100,RSSIu1))
90                  c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor) VALUES (?,
                    ?, ?)",(time.strftime("%Y/%m/%d %H:%M:%S"),101,RSSId1))
91                  conect_bd.commit()
92
93  ############################## DADOS DE A/Ds E IOs
    ##############################################
94                  for l in range(16,51,3):
95                      tipo_sensor = int(line[l])
96                      adh = int(line[l+1])
97                      adl = int(line[l+2])
98                      AD = adh * 256 + adl
99                      ## print ('tipo',tipo_sensor,'valor',AD)  ##Debug
100                     if (tipo_sensor > 4):
101                         print (' Tipo do sensor = ',tipo_sensor, ' ,Valor do sensor
                            = ',AD) # Apresentaçcao no
                            shell
102                         c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor)
                            VALUES (?, ?, ?)",(time.strftime("%Y/%m/%d
                            %H:%M:%S"),tipo_sensor,AD))
103                         conect_bd.commit()
104                     if (tipo_sensor == 1):
105                         controle[k] = Anser
106                         print (' Tipo do sensor = ',tipo_sensor, ' ,Status do sensor
                            = ',adh,'ID I/O = ',adl) # Apresentaçcao no
                            shell
107                         c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor,
                            IDCS) VALUES (?, ?, ?, ?)",(time.strftime("%Y/%m/%d
                            %H:%M:%S"),tipo_sensor,adh,adl))
108                         conect_bd.commit()
109                     ############### Caso haja necessidade de conta#####
110                     if (tipo_sensor == 2):
111                        variavel= AD*0.0032 * 2
112                        print (' Tipo do sensor = ',tipo_sensor, ' ,Valor do sensor =
                           ',variavel,'V') # Apresentaçcao no
                           shell
113                        c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor)
                           VALUES (?, ?, ?)",(time.strftime("%Y/%m/%d
                           %H:%M:%S"),tipo_sensor,variavel))
114                        conect_bd.commit()
115                     if (tipo_sensor == 3):
116                        variavel= AD/100 * 1.12 + 0.8
117                        print (' Tipo do sensor = ',tipo_sensor, ' ,Valor do sensor =
                           ',variavel,'C') # Apresentaçcao no
```

```
                        shell
118                     c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor)
                        VALUES (?, ?, ?)",(time.strftime("%Y/%m/%d
                        %H:%M:%S"),tipo_sensor,variavel))
119                     conect_bd.commit()
120
121                 if (tipo_sensor == 4):
122                     variavel= AD/100*0.67-0.44
123                     print (' Tipo do sensor = ',tipo_sensor, ' ,Valor do sensor =
                        ',variavel,'%') # Apresentaçcao no
                        shell
124                     c.execute("INSERT INTO Sensor"+Anser+"(Data, Tipo, Valor)
                        VALUES (?, ?, ?)",(time.strftime("%Y/%m/%d
                        %H:%M:%S"),tipo_sensor,variavel))
125                     conect_bd.commit()
126
127                 ########################################################
128             if line[4] == 2:
129                 print ('Sensor',Anser,' vai dormir(Sleep)')
130             if line[4] == 4:
131                 print ('Sensor',Anser,' vai dormir(Sleep2)')
132             if mac0 == 1:
133                 time.sleep(mac1) # Intervalo entre medidas
134             if mac0 == 3:
135                 time.sleep(mac1*15) # Intervalo entre medidas
136         else:
137             print ('Sensor',str(sensor+k),' nao respondeu ')
138             ser.flushInput()
139             time.sleep(1) # Intervalo de time out
140
141     print ('Fim da Execucao')  # escreve na tela
142
143
144
145
146 # In[4]:
147
148 def aux(i):
149     num_input = i
150     return num_input
151
152
153 # ## Porta de Comunicação ##
154 # Por ser uma conexão externa sera necessario estabelecer uma porta com para a
    conexão (No *windows*: Portas COM)
155
156 # In[5]:
157
158 ################################## Configura a serial
    ######################################################
159 # para COM# o numero que se coloca eh n-1 no primeiro parametrso. Ex COM9  valor 8
160 print('Porta(s) Ativa(s):')
161 for numero,portas_ativas in verifica_portas():
162     print('                %s' %portas_ativas)
163 finished="false"
164 while (finished != "true"):
165     finished="true"
166     n_serial = input('Digite o numero da serial desejada e pressione enter:') #seta
    a serial
167     n_serial1 = int(n_serial) - 1
168     state=1
169     for numero,portas_ativas in verifica_portas():
170         if n_serial1 == numero:
171             state= 0
172     if state == 1:
173         print('Serial invalida')
174         finished = "false"
175 ser = serial.Serial(n_serial1, 9600, timeout=1,parity=serial.PARITY_NONE) # seta
    valores da serial
176
177
178 # ## Protocolo Radiuino ##
```

```
179
180 # In[6]:
181
182 while True:
183     try:
184
185         ############################ Imprime na tela o menu
            ################################################
186         print ('Escolha uma das opcoes abaixo')
187         print ('[   1  ] --> Realizar medidas:')
188         print ('[Outros] --> Para sair:')
189         Opcao = input('Digite a opção e pressione enter: ')
190         ser.flushInput()# Limpa o buffer da serial
191
192         ########################### DADOS DOS SENSORES E QNTD DE MEDIDAS
            ###################################
193         if Opcao == "1":
194         ########################### CONFIGURAÇÃO DO BD
            ###################################
195             finished="false"
196             while (finished != "true"):
197                 finished = "false"
198                 nome_bd = input('Digite o nome do Aquivo do Banco de dados(no maximo
                    10 caractere):')
199                 if (len(nome_bd) <= 10):
200                     finished = "true"
201                 else:
202                     print('Numero maximo de caracteres é 10, tente denovo:')
203             conect_bd = sqlite3.connect(nome_bd+'.db')
204             c = conect_bd.cursor()
205
206             ##############################
207             finished="false"
208             while (finished != "true"):
209                 finished = "false"
210                 print ('Função Sleep2')
211                 sleep2 = int(input('Digite [ 1 ]--> Ativar [ 2 ]--> Desativar:'))
212
213
214                 if sleep2 == 1:
215                     mac0 = 3;
216                     print ('Os tempos são múltiplos de 15 segundos')
217                     print ('Ex. enviar o valor "1" equivale a 15 segundos, o valor
                        "2" 30 segundos, etc...')
218                     print ('[tempo máximo 63,75 minutos --> correspondente a 255] ')
219                     mac1 = int(input(' tempo de sleep (T1):'))
220                     mac2 = int(input(' tempo de espera em Escuta (T2):'))
221                     mac3 = int(input(' tempo de soneca (T3):'))
222                     finished = "true"
223                 elif sleep2 == 2:
224                     mac0 = 1;
225                     mac1 = int(input('Digite o intervalo das medidas(em segundos,
                        maximo 255): '))
226                     mac2 = 0;
227                     mac3 = 0;
228                     finished = "true"
229                 else:
230                     print('Opção invalida')
231
232
233             num_medidas = int(input('Digite o número de medidas e pressione enter:
                '))
234             qtd_sensor  =int(input('Digite a quantidade de nó(s) sensor(es) e
                pressione enter: '))
235             if qtd_sensor == 1:
236                 ID_sensor = int(input('Digite o ID do nó sensor e pressione enter:
                    ')) # ID Sensor, Byte[8]
237             else:
238                 ID_sensor = int(input('Digite o ID do primeiro nó sensor e pressione
                    enter: '))# ID Sensor, Byte[8]
239             realizar_medidas(num_medidas,qtd_sensor,ID_sensor,mac0,mac1,mac2,mac3)
240             conect_bd.close()
```

```
241             else:
242                 ser.flushInput()
243                 ser.close() # fecha a porta COM
244                 print ('Fim da Execucao')  # escreve na tela
245                 break
246         except KeyboardInterrupt:
247                 ser.flushInput()
248                 ser.close() # fecha a porta COM
249                 break
250
```

8.2 APÊNDICE2 – FIRMWARE NSS

8.2.1 Aba principal

```
1  // Radiuino_Sensor : firmware para os nos Sensores da rede
2
3  // Mais informacoes em www.radiuino.cc
4  // Copyright (c) 2015
5  // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
   Ferreira
6  // Versao 2.2: 20/01/2015
7
8  // Este arquivo e parte da plataforma Radiuino
9  // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
   dentro dos termos da Licenca Publica Geral Menor GNU
10 // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
   (na sua opniao) qualquer futura versao.
11 // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
   GARANTIA; sem uma garantia implicita
12 // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
   Publica Geral Menor GNU para maiores detalhes.
13 // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
   este programa, se nao, escreva para a Fundacao
14 // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
   USA
15
16 // This library is free software; you can redistribute it and/or modify it under the
   terms of the GNU Lesser General Public License
17 // as published by the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version. This library
18 // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
   without even the implied warranty of MERCHANTABILITY
19 // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
   for more details. You should have received a copy
20 // of the GNU Lesser General Public License along with this library; if not, write
   to the Free Software Foundation, Inc., 51 Franklin St,
21 // Fifth Floor, Boston, MA  02110-1301  USA
22
23 #include <RADIUINO.h>
24 #include <EEPROM.h>
25 #include <SPI.h>
26 #include <DHT.h>
27 #include "Headers.h"
28
29 DHT dht(7, DHT22);          /* parametos do dht (DHTPIN, DHTTYPE) */
30
31 #define FIRMWARE_VERSION   1.0     /* Versao 1.0 Radiuino */
32 byte int_rx = 0;                   /* Inicializacao da interrupcao de recepcao de
   pacotes - O hardware gera uma interrup□□o no come□o que nao deve ser tratada */
33 byte int_buff = 0;                 /* Inicializacao da interrupcao de buffer
   overflow - O hardware gera uma interrupcao no come□o que nao deve ser tratada */
34 int Temperatura, Umidade, AD0;
35 /**
36  * Configura o Arduino. Executado uma unica vez no inicio do firmware.
37  */
38 void setup()
39 {
40   /* Inicializacao da camada F□sica */
41   Phy.initialize();
42
43   /* Inicializacao da camada de Controle de Acesso ao Meio */
44   Mac.initialize();
45
46   /* Inicializacao da camada de Rede */
47   Net.initialize();
48
49   /* Inicializacao da camada de Transporte */
50   Transp.initialize();
51
52   /* Inicializacao da camada de Aplicacao */
53   App.initialize();
54
55   /* Anexa as funcoes de interrupcao de recepcao de pacotes e de estouro de buffer
     de recepcao */
56   attachInterrupt(0, IntReceiveData, RISING);
57   attachInterrupt(1, IntBufferOverflow, RISING);
58   pinMode(GDO0, INPUT);
```

```
59
60
61    /* Inicializa o Timer1 e configura o periodo para 1 segundo */
62    Timer1.initialize(1000000);
63
64    /* Anexa a funcao de interrupcao de estouro do Timer1 */
65    Timer1.attachInterrupt(IntTimer1);
66
67    /* Escreve mensagem de inicializacao */
68    Serial.print("Radiuino! Sensor");
69    pinMode(0, INPUT);
70    pinMode (7, INPUT); /* dados do sensor DHT */
71    pinMode (8, OUTPUT); /* Alimentacao do sensor DHT */
72    digitalWrite(8, LOW); /*incializa em low para economizar energia */
73    dht.begin();
74
75  }
76
77  /**
78   * Laco de execucao do Arduino. Executado continuamente.
79   */
80  void loop()
81  {
82
83    digitalWrite(8, HIGH);   // Alimenta o sensor
84      delay(2000);
85
86      float h = dht.readHumidity();
87      float t = dht.readTemperature();
88
89      // check if returns are valid, if they are NaN (not a number) then something
      went wrong!
90      if (isnan(t) || isnan(h))
91      {
92          t = 99;
93          h= 100;
94      }
95      else
96      {
97         delay(1000);
98         Temperatura = int(t)*100;
99         Umidade = int(h)*100;
100     }
101     digitalWrite(8, LOW);
102
103     /* Verifica se deve entrar em modo Sleep */
104     verifySleepEntering();
105
106 }
107
108 /**
109  * Trata o pacote recebido pela rede sem fio.
110  */
111 void IntReceiveData()
112 {
113
114   /* Se for a primeira vez que a interrupcao  executada */
115   if (int_rx == 0)
116   {
117     int_rx = 1;
118     return;
119   }
120
121   /* Reenvia toda informacao recebida da rede para o PC */
122   if ( digitalRead(GDO0) == HIGH )
123   {
124     /* Recebe os dados do RF */
125     Phy.receive(&g_pkt);
126   }
127
128   return;
129 }
130
```

```
131  /**
132   * Trata o estouro do buffer de recepcao.
133   */
134  void IntBufferOverflow()
135  {
136    /* Se for a primeira vez que a interrupcao □ executada */
137    if (int_buff == 0) {
138      int_buff = 1;
139      return;
140    }
141
142    /* Esvazia o buffer de recepcao e vai para o estado de RX */
143    cc1101.Strobe(CC1101_SFRX);
144    cc1101.Strobe(CC1101_SRX);
145
146    return;
147  }
148
149  /**
150   * Trata o estouro do per□odo do Timer1.
151   */
152  void IntTimer1()
153  {
154    /* Aqui podem ser colocadas tarefas peri□dicas. O per□odo do Timer1 □ de 1 segundo
     por padr□o.
155     * Mas pode ser mudado no setup() */
156
157    return;
158  }
159
160  /**
161   * Realiza uma operacao Atomica para verificar se deve entrar no modo Sleep ou se
     deve sair do modo Sleep
162   */
163  void verifySleepEntering ( void ) {
164    ATOMIC_BLOCK(ATOMIC_FORCEON)
165    {
166      if (Mac.time_to_sleep > 0) {
167
168        /* Aguarda o fim da tranmissao do ACK SLEEP */
169        delayMicroseconds(500);
170        while(Phy.txFifoFree() != CC1101_FIFO_SIZE);
171
172        /* Entra em modo Sleep */
173        Phy.lowPowerOn();
174
175      }
176      else if (Mac.time_to_sleep == 0) {
177
178        /* Saindo do modo de Sleep */
179        cc1101.Strobe(CC1101_SIDLE);
180        delay(1);
181        cc1101.Strobe(CC1101_SFTX);
182        cc1101.Strobe(CC1101_SFRX);
183        cc1101.Strobe(CC1101_SRX);
184
185        Mac.time_to_sleep = -1;
186        Mac.last_wakeup = millis();
187        /* Habilita o conversor AD */
188        _SFR_BYTE(ADCSRA) |= _BV(ADEN);
189      }
190      if ((Mac.flag_sleep2)&&(Mac.time_to_sleep <  0)){
191        if( (millis() - Mac.last_wakeup) > (Mac.t2 * 1000 )){
192           Mac.time_to_sleep = Mac.t3;
193        }
194      }
195    }
196  }
197
198  /**
199   * Interrupcao de WatchDog. Executada sempre que o contador de WatchDog estoura
200   */
201  ISR(WDT_vect) {
```

```
202
203     /* Verifica se precisamos decrementar o contador de Sleep */
204     if (Mac.time_to_sleep > 0) {
205       Mac.time_to_sleep--;
206       return;
207     }
208
209   }
210
```

8.2.2 Aba Headers.h

```
1   // Header.h : cabecalhos de classes
2
3   // Mais informacoes em www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #ifndef HEADERS_H
24  #define HEADERS_H 1
25
26  /**
27   * Estrutura do pacote a ser transmitido e recebido. O pacote possui 52 bytes, sendo
     4 bytes de cabecalho de camada fisica (Phy), 4 bytes de cabecalho
28   * de camada de Controle de Acesso ao Meio (MAC), 4 bytes de cabecalho de camada de
     Rede (Net), 4 bytes de cabecalho de camada de Transporte (Transp) e
29   * e 4 bytes de cabecalho de camada de Aplicacao (App). Os 36 bytes restantes sao
     reservados para payload de AD e IO.
30   */
31  typedef struct
32  {
33    /* Cabecalho da camada Fisica (Phy) */
34    byte PhyHdr[4];
35
36    /* Cabecalho da camada de Controle de Acesso ao Meio (MAC) */
37    byte MACHdr[4];
38
39    /* Cabecalho da camada de Rede (Net) */
40    byte NetHdr[4];
41
42    /* Cabecalho da camada de Transporte (Transp) */
43    byte TranspHdr[4];
44
45    /* Bytes o payload de AD */
46    byte AD0[3];
47    byte AD1[3];
48    byte AD2[3];
49    byte AD3[3];
50    byte AD4[3];
51    byte AD5[3];
52
53    /* Bytes do payload de IO */
54    byte IO0[3];
55    byte IO1[3];
56    byte IO2[3];
57    byte IO3[3];
58    byte IO4[3];
```

```
59    byte IO5[3];
60
61  } packet;
62
63  packet g_pkt;
64
65  /**
66   * Classe de camada de Aplicacao
67   */
68
69  #define AD0_PIN  0
70  #define AD1_PIN  1
71  #define AD2_PIN  2
72  #define AD3_PIN  3
73  #define AD4_PIN  4
74  #define AD5_PIN  5
75
76  #define IO0_PIN  4
77  #define IO1_PIN  5
78  #define IO2_PIN  6
79  #define IO3_PIN  7
80  #define IO4_PIN  8
81  #define IO5_PIN  9
82
83  class APP
84  {
85    public:
86      APP(void);
87      inline void initialize(void);
88      inline void send(packet * pkt);
89      inline void receive(packet * pkt);
90
91    private:
92      int IO_0, IO_1, IO_2, IO_3, IO_4, IO_5;
93
94  };
95
96  /* Objeto de acesso a classe da camada de Aplicacao */
97  extern APP App;
98
99  /**
100  * Classe de camada de Transporte
101  */
102 class TRANSP
103 {
104   public:
105     TRANSP(void);
106     inline void initialize(void);
107     inline void send(packet * pkt);
108     inline void receive(packet * pkt);
109
110   private:
111
112 };
113
114 /* Objeto de acesso a classe da camada de Transporte */
115 extern TRANSP Transp;
116
117 /**
118  * Classe de camada de Rede
119  */
120 class NET
121 {
122   public:
123     NET(void);
124     inline void initialize(void);
125     inline void send(packet * pkt);
126     inline void receive(packet * pkt);
127     void swapAddresses(packet * pkt);
128
129     byte my_addr;
130   private:
131
```

```
132  };
133
134  /* Objeto de acesso a classe da camada de Rede */
135  extern NET Net;
136
137  /**
138   * Classe de camada de Controle de Acesso ao Meio
139   */
140  #define  SLEEP_MSG  1
141  #define  SLEEP_ACK  2
142  #define  SLEEP2_MSG  3
143  #define  SLEEP2_ACK  4
144  class MAC
145  {
146    public:
147      MAC(void);
148      inline void initialize(void);
149      inline void send(packet * pkt);
150      inline void receive(packet * pkt);
151
152      volatile int time_to_sleep;
153      volatile unsigned long t2;
154      volatile int t3;
155      volatile boolean flag_sleep2;
156      volatile unsigned long last_wakeup;
157    private:
158
159  };
160
161  /* Objeto de acesso a classe da camada de Controle de Acesso ao Meio */
162  extern MAC Mac;
163
164  /**
165   * Classe de camada Fisica
166   */
167  #define BUFFLEN  CC1101_PACKT_LEN
168  byte serialData[BUFFLEN + 1];
169
170  class PHY
171  {
172    public:
173      PHY(void);
174      inline void initialize();
175      inline void send(packet * pkt);
176      inline int receive(packet * pkt);
177
178      void sendSerial(packet * pkt);
179      void receiveSerial(void);
180
181      byte txFifoFree(void);
182      void setChannel(byte channel);
183      void lowPowerOn(void);
184      void setPower(byte power);
185      void setFreqOffset(byte freq_offset);
186
187      boolean carrierSense(void);
188
189      byte power;                 /* Potencia */
190      byte channel;               /* Canal */
191      byte freq_offset;           /* Offset de frequencia */
192      int serial_baudrate;        /* Serial baudrate */
193
194
195
196    private:
197      int initCC1101Config(void);
198      void configWatchdog(int time);
199      void strobe_idle_wait(void);
200  };
201
202  /* Objeto de acesso a classe da camada Fisica */
203  extern PHY Phy;
204
```

```
205  /* Configuracao de registradores do CC1101. Obtidos atraves do SmartRF Studio 7 */
206  // Deviation = 4.760742
207  // Base frequency = 914.999969
208  // Carrier frequency = 914.999969
209  // Channel number = 0
210  // Carrier frequency = 914.999969
211  // Modulated = true
212  // Modulation format = GFSK
213  // Manchester enable = false
214  // Sync word qualifier mode = 30/32 sync word bits detected
215  // Preamble count = 4
216  // Channel spacing = 199.951172
217  // Carrier frequency = 914.999969
218  // Data rate = 9.59587
219  // RX filter BW = 203.125000
220  // Data format = Normal mode
221  // CRC enable = true
222  // Device address = 0
223  // Address config = No address check
224  // CRC autoflush = false
225  // PA ramping = false
226  // TX power = 10
227  const byte CC1101_registerSettings[CC1101_NR_OF_CONFIGS][CC1101_NR_OF_REGISTERS]
     PROGMEM = {
228  {
229    0x04,  // IOCFG2  GDO2 Output Pin Configuration
230    0x07,  // IOCFG0  GDO0 Output Pin Configuration
231    0x47,  // FIFOTHR  RX FIFO and TX FIFO Thresholds
232    0x34,  // PKTLEN  Packet Length
233    0x04,  // PKTCTRL1  Packet Automation Control
234    0x04,  // PKTCTRL0  Packet Automation Control
235    0x00,  // ADDR  Device Address
236    0x00,  // CHANNR  Channel Number
237    0x06,  // FSCTRL1  Frequency Synthesizer Control
238    0x00,  // FSCTRL0  Frequency Synthesizer Control
239    0x23,  // FREQ2  Frequency Control Word, High Byte
240    0x31,  // FREQ1  Frequency Control Word, Middle Byte
241    0x3B,  // FREQ0  Frequency Control Word, Low Byte
242    0xF7,  // MDMCFG4  Modem Configuration
243    0x83,  // MDMCFG3  Modem Configuration
244    0x03,  // MDMCFG2  Modem Configuration
245    0xA2,  // MDMCFG1  Modem Configuration
246    0x3B,  // MDMCFG0  Modem Configuration
247    0x31,  // DEVIATN  Modem Deviation Setting
248    0x18,  // MCSM0  Main Radio Control State Machine Configuration
249    0x16,  // FOCCFG  Frequency Offset Compensation Configuration
250    0x6C,  // BSCFG  Bit Synchronization Configuration
251    0x43,  // AGCCTRL2  AGC Control
252    0x40,  // AGCCTRL1  AGC Control
253    0x91,  // AGCCTRL0  AGC Control
254    0x56,  // FREND1  Front End RX Configuration
255    0x10,  // FREND0  Front End TX Configuration
256    0xE9,  // FSCAL3  Frequency Synthesizer Calibration
257    0x2A,  // FSCAL2  Frequency Synthesizer Calibration
258    0x00,  // FSCAL1  Frequency Synthesizer Calibration
259    0x1F,  // FSCAL0  Frequency Synthesizer Calibration
260    0x59,  // FSTEST  Frequency Synthesizer Calibration Control
261    0x81,  // TEST2  Various Test Settings
262    0x35,  // TEST1  Various Test Settings
263    0x09,  // TEST0  Various Test Settings
264  }
265  };
266
267  const byte CC1101_paTable[CC1101_NR_OF_CONFIGS][CC1101_PA_TABLESIZE] PROGMEM ={
268  // -30  -20   -15  -10   0   5   7   10
269    {0x03,0x0E,0x1E,0x27,0x8E,0x84,0xCC,0xC3},    // Configuracao 0
270  };
271
272  #endif
273
```

8.2.3 Aba _1_Phy

```
1   // Phy : classe da camada F?sica
2
3   // Mais informacoes em www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada Fisica.
27   */
28  PHY::PHY()
29  {
30    /********** AJUSTE DE POT?NCIA **********/
31    /* A potencia de transmissao pode ser escolhida entre 8 valores possiveis (0 a 7).
32     * Abaixoa tabela que relaciona o numero com a potencia de transmissao.
33     * Em geral a potencia eh ajustada para o maximo de 7 (10 dBm).
34     * | -30 | -20 | -15 | -10 |   0 |   5 |   7 |  10 |   - Potencia em dBm
35     * |   0 |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   - Numero que deve ser
       colocado na potencia
36     * !!! ATENCAO !!!: Potencias maiores que 0dBm podem fazer com que as placas
       percam a comunicacao quando estao muito proximas. Recomendamos posiciona-las com
       uma distancia minima de 1 metro. */
37    power = 7;
38
39    /********** CANAL DE COMUNICACAO **********/
40    /* Pode ser escolhido de 0 a 65.
41    /* Os canais estao espacados de 200kHz e dispostos na faixa de 915 a 928 MHz */
42    channel = 3;
43
44    /********** AJUSTE DE OFFSET **********/
45    /* A diferenca de frequencia dos cristais de cada placa deve ser compensada.
46     * Este valor esta escrito em cada placa do BE900. */
47    freq_offset = 0x11;
48
49    /********** Serial Baudrate **********/
50    serial_baudrate = 9600;
51  }
52
53  /**
54   * Inicializa a camada Fisica.
55   */
56  void PHY::initialize(void)
57  {
58    /* Configurando o serial baudrate */
```

```
59    Serial.begin(serial_baudrate);
60
61    //* Inicializa o transceptor CC1101 */
62    cc1101.PowerOnStartUp();
63
64    /* Inicializa a configuracao do transceptor CC1101 */
65    initCC1101Config();
66
67    /* Configura o canal a ser usada */
68    setChannel(channel);
69
70    /* COnfigura a potencia a ser usada */
71    setPower(power);
72
73    /* Configura o offset de frequencia */
74    setFreqOffset(freq_offset);
75
76    /* Configura o timeout do WatchDog para 1 segundo */
77    configWatchdog(6);
78
79
80
81  }
82
83  /**
84   * Recebe dados da porta serial.
85   */
86  void PHY::receiveSerial(void)
87  {
88    byte len;             /* Tamanho do dado recebido pela porta serial */
89    byte fifoSize = 0;    /* Tamanho do espaco livre no FIFO de TX */
90
91    static byte pos = 0;  /* Total de bytes recebidos pela porta serial */
92
93    /* Le a porta serial e incrementa o total de bytes recebidos */
94    len = Serial.available() + pos;
95
96    /* Processa no maximo BUFFLEN bytes */
97    if (len > BUFFLEN )
98    {
99      len = BUFFLEN;
100   }
101
102   /* Verifica quanto espaco temos no FIFO de TX */
103   fifoSize = Phy.txFifoFree();   /* O fifoSize deve ter o numero de bytes
      atualmeente livre no FIFO de TX */
104
105   /* Reinicializa as variaveis e sai da funcao */
106   if ( fifoSize <= 0)
107   {
108     Serial.flush();
109     pos = 0;
110     return;
111   }
112
113   /* Evita estourar o FIFO de TX */
114   if (len > fifoSize)
115   {
116     len = fifoSize;
117   }
118
119   /* Finalmente escreve os dados lidos da serial no FIFO de TX */
120   for (byte i = pos; i < len; i++)
121   {
122     serialData[i] = Serial.read();  /* serialData eh o nosso buffer global */
123   }
124
125   /* Atraso de 1 milissegundo */
126   delayMicroseconds(1000);
127
128   /* Verifica se existem mais dados para receber */
129   if ((Serial.available() > 0)  && (len < CC1101_PACKT_LEN))
130   {
```

```
131       pos = len;  /* Mantem a quantidade de bytes ja recebidos e espera pela proxima
          entrada nessa funcao */
132       return;
133     }
134
135     if (len == sizeof(packet))
136     {
137       /* Envia a mensagem recebida pelo RF */
138       Phy.send((packet *)serialData);
139       /* O buffer da serial esta livre novamente */
140       pos = 0;
141     }
142     else
143     {
144       Serial.flush();
145       pos = 0;
146       return;
147     }
148   }
149
150   /**
151    * Transmite dados pela porta serial.
152    */
153   void PHY::sendSerial(packet * pkt)
154   {
155     /* Escreve o pacote inteiro na porta serial */
156     Serial.write((byte *)pkt, sizeof(packet));
157   }
158
159   /**
160    * Le o espaco disponivel no FIFO de TX.
161    */
162   byte PHY::txFifoFree(void)
163   {
164     byte size;
165
166     cc1101.Read(CC1101_TXBYTES, &size);
167
168     /* Trata um possivel underflow to FIFO de TX */
169     if (size >= 64)
170     {
171       cc1101.Strobe(CC1101_SFTX);
172       cc1101.Read(CC1101_TXBYTES,&size);
173     }
174
175     return (CC1101_FIFO_SIZE - size);
176   }
177
178   /**
179    * Ajusta o canal a ser usado.
180    */
181   void PHY::setChannel(byte channel)
182   {
183     cc1101.Write(CC1101_CHANNR, channel);
184   }
185
186   /**
187    * Ajusta a potencia a ser usada.
188    */
189   void PHY::setPower(byte power)
190   {
191     cc1101.setPA(0, power);
192   }
193
194   /**
195    * Ajusta o offset de frequencia.
196    */
197   void PHY::setFreqOffset(byte freq_offset)
198   {
199     cc1101.Write(CC1101_FSCTRL0, freq_offset);
200   }
201
202   /**
```

```
203    * Inicializa a configuracao do CC1101.
204    */
205   int PHY::initCC1101Config(void)
206   {
207     /* Carrega a primeira configuracao (pode ser inserida mais de uma configuracao no
       codigo) */
208     cc1101.Setup(0);
209
210     /* Configura o endereco do radio */
211     cc1101.Write(CC1101_ADDR, Net.my_addr);
212
213     /* Configura a potencia para a maxima possivel (7) */
214     cc1101.setPA(0, 7);
215
216     /* Coloca o CC1101 no estado de RX */
217     cc1101.Strobe(CC1101_SIDLE);
218     delay(1);
219     cc1101.Write(CC1101_MCSM1 ,   0x0F );
220     cc1101.Strobe(CC1101_SFTX);
221     cc1101.Strobe(CC1101_SFRX);
222     cc1101.Strobe(CC1101_SRX);
223
224     return OK;
225   }
226
227   /**
228    * Entra em estado de Low Power mode
229    */
230   void PHY::lowPowerOn(void){
231
232     /* Colocando o CC1101 no estado de SLEEP */
233     cc1101.Strobe(CC1101_SIDLE);
234     cc1101.Strobe(CC1101_SPWD);
235
236     /* Desabilita o conversor AD */
237     _SFR_BYTE(ADCSRA) &= ~_BV(ADEN);
238
239     /* Configura as portas de saida como entrada */
240     pinMode (IO0_PIN, INPUT);
241     pinMode (IO1_PIN, INPUT);
242     pinMode (IO2_PIN, INPUT);
243     pinMode (IO3_PIN, INPUT);
244     pinMode (IO4_PIN, INPUT);
245     pinMode (IO5_PIN, INPUT);
246
247     /* Configura o modo Sleep do ATMega */
248     set_sleep_mode(SLEEP_MODE_PWR_DOWN);
249
250     cli(); sleep_enable();          /* Habilita o bit de Sleep no registrador MCUCR */
251
252     /* Desliga os modulos do ATMega */
253     power_adc_disable();
254     power_spi_disable();
255     power_timer0_disable();
256     power_usart0_disable();
257     power_timer0_disable();
258     power_timer1_disable();
259     power_timer2_disable();
260     power_twi_disable();
261
262     /* Coloca o ATMega em modo Sleep efetivamente */
263     sei(); sleep_mode();
264
265     /**** O PROGRAMA CONTINUA AQUI DEPOIS DE ACORDAR ****/
266     sleep_disable();         /* Desabilita o bit de Sleep */
267
268     /* Configura as portas de saida como saida */
269     pinMode (IO0_PIN, OUTPUT);
270     pinMode (IO1_PIN, OUTPUT);
271     pinMode (IO2_PIN, OUTPUT);
272     pinMode (IO3_PIN, OUTPUT);
273     pinMode (IO4_PIN, OUTPUT);
274     pinMode (IO5_PIN, OUTPUT);
```

```
275
276     /* Habilita o conversor AD */
277     _SFR_BYTE(ADCSRA) |= _BV(ADEN);
278
279     /* Liga todos os modulos do ATMega */
280     power_all_enable();
281
282   }
283
284   /**
285    * Retorna o status de presenca de portadora no canal (Carrier Sense)
286    * return   1 se o canal esta ocupado, 0 se o canal esta livre
287    */
288   boolean PHY::carrierSense(void)
289   {
290     byte cs;
291
292     /* O status da portadora eh lido no registrador PKTSTATUS */
293     cc1101.Read(CC1101_PKTSTATUS,&cs);
294     /* O bit de Carrier Sense eh o bit 6 */
295     cs &= 0x40;
296
297     if (cs)
298       return true;
299     else
300       return false;
301
302   }
303
304   /**
305    * Configura o valor do timeout do WatchDog
306    * 0=16ms, 1=32ms, 2=64ms, 3=128ms, 4=250ms, 5=500ms, 6=1sec, 7=2sec, 8=4sec, 9=8sec
307    */
308   void PHY::configWatchdog(int time) {
309
310     byte value;
311
312     if (time > 9 ) time = 9;
313     value = time & 7;
314     if (time > 7) value |= (1<<5);
315     value |= (1<<WDCE);
316
317     /* Habilita a interrup??o de WatchDog no Satus Register */
318     MCUSR &= ~(1<<WDRF);
319
320     /* Configura as flags do registrador de WatchDog */
321     WDTCSR |= (1<<WDCE) | (1<<WDE);
322
323     /* Configura o valor do timeout */
324     WDTCSR = value;
325
326     /* Habilita a interrup??o do WatchDog */
327     WDTCSR |= _BV(WDIE);
328
329   }
330
331   /**
332    *    Strobe a command in order to the CC1101 go to the IDLE state
333    */
334
335   void PHY::strobe_idle_wait( void )
336   {
337     byte current_state;
338
339     /* Strobe to IDLE state */
340     cc1101.Strobe(CC1101_SIDLE);
341
342     do
343     {
344       /* Read current state */
345       cc1101.Read( CC1101_MARCSTATE, &current_state);
346
347     /* Until in IDLE state */
```

Aba _2_MAC

```
1   // MAC : classe da camada de Controle de Acesso ao Meio
2
3   // Mais informacoes em www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada de Controle de Acesso ao Meio.
27   */
28  MAC::MAC()
29  {
30  }
31
32  /**
33   * Inicializa a camada de Controle de Acesso ao Meio.
34   */
35  void MAC::initialize(void)
36  {
37    time_to_sleep = -1;
38
39    /* Seta a flag do Sleep2 para False */
40    flag_sleep2 = false;
41    last_wakeup = millis();
42  }
43
44  /**
45   * Envia o pacote para a camada inferior
46   */
47  inline void MAC::send(packet * pkt)
48  {
49    unsigned long starttime = millis();
50
51    /* Aguarda enquanto o canal esta ocupado. Espera no maximo 100 ms */
52    while(Phy.carrierSense() && ((millis() - starttime) < 100));
53
54    /* Envia para a camada inferior */
55    Phy.send(pkt);
56
57    return;
58  }
59
60  /**
61   * Recebe o pacote da camada inferior
```

```
 */
inline void MAC::receive(packet * pkt)
{
  /* Se a mensagem e do tipo SLEEP */
  if (pkt->MACHdr[0] == SLEEP_MSG) {

    /* Calcula o tempo total para dormir */
    time_to_sleep = 256 * pkt->MACHdr[2] + pkt->MACHdr[1];

    /* Retorna um pacote de SLEEP ACK para o emissor */
    pkt->MACHdr[0] = SLEEP_ACK;

    Net.receive(pkt);
  }

  else if (pkt->MACHdr[0] == SLEEP2_MSG) {

    /* Calcula o tempo total para dormir */
    time_to_sleep = pkt->MACHdr[1]*15 ;
    t2 = pkt->MACHdr[2]*15 ;
    t3 = pkt->MACHdr[3]*15 ;

    /* Retorna um pacote de SLEEP ACK para o emissor */
    pkt->MACHdr[0] = SLEEP2_ACK;

    /* Seta a flag do Sleep2 para True */
    flag_sleep2 = true;

    Net.receive(pkt);
  }

  else {
    /* Seta a flag do Sleep2 para False */
    flag_sleep2 = false;
    Net.receive(pkt);
  }

  return;
}

/* Instanciacao do objeto de acesso a classe da camada de Controle de Acesso ao Meio
*/
MAC Mac = MAC();

```

8.2.4 Aba _3_Net

```
1   // NET : classe da camada de Rede
2
3   // Mais informacoes em www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
    Ferreira
6   // Vers□o 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca P□blica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na vers□o 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa □ distribuido na esperan□a que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licen□a
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Funda□ao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada de Rede.
27   */
28  NET::NET()
29  {
30    my_addr = 1;    /* Endere□o */
31  }
32
33  /**
34   * Inicializa a camada de Controle de Acesso ao Meio.
35   */
36  void NET::initialize(void)
37  {
38  }
39
40  /**
41   * Realiza a troca de endere□os Origem e Destino
42   */
43  void NET::swapAddresses(packet * pkt)
44  {
45    /* Troca os endere□os de destino e origem para a retranmiss□o dos pacotes */
46    pkt->NetHdr[0] = pkt->NetHdr[2];
47    pkt->NetHdr[2] = Net.my_addr;
48  }
49
50  /**
51   * Envia o pacote para a camada inferior
52   */
53  inline void NET::send(packet * pkt)
54  {
55    /* Envia para a camada inferior */
56    Mac.send(pkt);
57
58    return;
59  }
60
61  /**
```

```
62    * Recebe o pacote da camada inferior
63    */
64   inline void NET::receive(packet * pkt)
65   {
66     /* Troca os endere□os de destino e origem para a retranmiss□o dos pacotes */
67     pkt->NetHdr[0] = pkt->NetHdr[2];
68     pkt->NetHdr[2] = Net.my_addr;
69
70     /* Envia para a camada superior */
71     Transp.receive(pkt);
72
73     return;
74   }
75
76   /* Instancia□□o do objeto de acesso □ classe da camada de Rede */
77   NET Net = NET();
78
```

8.2.5 Aba _4_Transp

```
1   // Tansp : classe da camada de Transporte
2
3   // Mais informacoes em www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
    Ferreira
6   // Vers□o 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada de Transporte.
27   */
28  TRANSP::TRANSP()
29  {
30  }
31
32  /**
33   * Inicializa a camada de Transporte.
34   */
35  void TRANSP::initialize(void)
36  {
37  }
38
39  /**
40   * Envia o pacote para a camada inferior
41   */
42  inline void TRANSP::send(packet * pkt)
43  {
44    /* Envia para a camada inferior */
45    Net.send(pkt);
46
47    return;
48  }
49
50  /**
51   * Recebe o pacote da camada inferior
52   */
53  inline void TRANSP::receive(packet * pkt)
54  {
55    static byte counter = 0;
56
57    /* Insere um contado no cabe□alho de transporte do pacote */
58    pkt->TranspHdr[0] = counter++;
59
60    /* Envia para a camada superior */
61    App.receive(pkt);
```

```
62
63      return;
64    }
65
66    /* Instancia□□o do objeto de acesso □ classe da camada de Transporte */
67    TRANSP Transp = TRANSP();
68
```

8.2.6 Aba _5_App

```
1  // APP : classe da camada de Aplica□□o
2
3  // Mais informacoes em www.radiuino.cc
4  // Copyright (c) 2015
5  // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli e Debora M
   Ferreira
6  // Versao 2.2: 20/01/2015
7
8  // Este arquivo e parte da plataforma Radiuino
9  // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
   dentro dos termos da Licenca Publica Geral Menor GNU
10 // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
   (na sua opniao) qualquer futura versao.
11 // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
   GARANTIA; sem uma garantia implicita
12 // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
   Publica Geral Menor GNU para maiores detalhes.
13 // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
   este programa, se nao, escreva para a Fundacao
14 // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
   USA
15
16 // This library is free software; you can redistribute it and/or modify it under the
   terms of the GNU Lesser General Public License
17 // as published by the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version. This library
18 // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
   without even the implied warranty of MERCHANTABILITY
19 // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
   for more details. You should have received a copy
20 // of the GNU Lesser General Public License along with this library; if not, write
   to the Free Software Foundation, Inc., 51 Franklin St,
21 // Fifth Floor, Boston, MA  02110-1301  USA
22
23 #include "Headers.h"
24
25 /**
26  * Construtor da camada de Aplica□□o.
27  */
28 APP::APP()
29 {
30 }
31
32 /**
33  * Inicializa a camada de Aplica□□o.
34  */
35 void APP::initialize(void)
36 {
37   // Faz com que todos os pinos de IO sejam sa□da
38   pinMode (7, INPUT);
39   pinMode (A0, INPUT);
40
41 }
42
43 /**
44  * Envia o pacote para a camada inferior
45  */
46 inline void APP::send(packet * pkt)
47 {
48   return;
49 }
50
51 /**
52  * Recebe o pacote da camada inferior
53  */
54 inline void APP::receive(packet * pkt)
55 {
56   int AD0, AD1, AD2, AD3, AD4, AD5;
57
58  // AD0 - pino 15 do BE900
59   AD0 = analogRead(A0); // tensao de bateria
60   pkt->AD0[0] = 2; // Pode ser utilizado para indicar o tipo de sensor no byte 16
61   pkt->AD0[1] = (byte) (AD0/256); // Valor inteiro no byte 17
```

```
62      pkt->AD0[2] = (byte) (AD0%256); // Resto da divis□o noo byte 18
63      // No computador para voltar no valor de 0 a 1023 deve ser feita a conta
        Valor=Inteiro*256+Resto
64
65
66      // AD1 - pino 13 do BE900
67      AD1 = Temperatura;
68      pkt->AD1[0] = 3; // Pode ser utilizado para indicar o tipo de sensor o byte 19
69      pkt->AD1[1] = (byte) (AD1/256); // Valor inteiro no byte 20
70      pkt->AD1[2] = (byte) (AD1%256); // Resto da divis□o noo byte 21
71
72
73      // AD2 - pino 12 do BE900
74      AD2 = Umidade;
75      pkt->AD2[0] = 4; // Pode ser utilizado para indicar o tipo de sensor o byte 22
76      pkt->AD2[1] = (byte) (AD2/256); // Valor inteiro no byte 23
77      pkt->AD2[2] = (byte) (AD2%256); // Resto da divis□o noo byte 24
78
79      // TRANSMITE PACOTE
80      Transp.send(pkt);
81
82      return;
83    }
84
85    /* Instancia□□o do objeto de acesso □ classe da camada de Aplica□□o */
86    APP App = APP();
```

8.3 APÊNDICE3 – FIRMWARE BASE

8.3.1 Aba principal

```
1   // Radiuino_Base : firmware para o no Base da rede
2
3   // Mais informacoes acesse www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli, Debora M.
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include <RADIUINO.h>
24  #include <EEPROM.h>
25  #include <SPI.h>
26
27  #include "Headers.h"
28
29  #define FIRMWARE_VERSION   2.2     /* Versao 2.2 Radiuino */
30  byte int_rx = 0;                   /* Inicializacao da interrupcao de recepcao de
    pacotes - O hardware gera uma interrupcao no comeco que nao deve ser tratada */
31  byte int_buff = 0;                 /* Inicializacao da interrupcao de buffer
    overflow - O hardware gera uma interrupcao no comeco que nao deve ser tratada */
32
33  /**
34     Configura o Arduino. Executado uma unica vez no inicio do firmware.
35  */
36  void setup()
37  {
38    /* Inicializacao da camada Fisica */
39    Phy.initialize();
40
41    /* Inicializacao da camada de Controle de Acesso ao Meio */
42    Mac.initialize();
43
44    /* Inicializacao da camada de Rede */
45    Net.initialize();
46
47    /* Anexa as funcoes de interrupcao de recepcao de pacotes e de estouro de buffer
      de recepcao */
48    attachInterrupt(0, IntReceiveData, RISING);
49    attachInterrupt(1, IntBufferOverflow, RISING);
50    pinMode(GDO0, INPUT);
51
52    /* Inicializa o Timer1 e configura o periodo para 1 segundo */
53    Timer1.initialize(1000000);
54
55    /* Anexa a funÃƒÂ¯Ã‚Â¿Ã‚Â½Ã‚Â¯Ã‚Â¿Ã‚Â½o de interrupcao de estouro do Timer1 */
56    Timer1.attachInterrupt(IntTimer1);
57
58    /* Escreve mensagem de inicializacao */
```

```
59      Serial.print("Radiuino! Base");
60
61    }
62
63    /**
64       Laco de execucao do Arduino. Executado continuamente.
65    */
66    void loop()
67    {
68      /* Aguarda pela recepcao de comandos da porta serial */
69      if (Serial.available() > 0)
70      {
71        Phy.receiveSerial();
72      }
73    }
74
75    /**
76       Trata o pacote recebido pela rede sem fio.
77    */
78    void IntReceiveData()
79    {
80      /* Se for a primeira vez que a interrupcao e executada */
81      if (int_rx == 0)
82      {
83        int_rx = 1;
84        return;
85      }
86
87      /* Reenvia toda informacao recebida da rede para o PC */
88      if ( digitalRead(GDO0) == HIGH ) {
89
90        /* Recebe os dados do RF */
91        if (Phy.receive(&g_pkt) == ERR)
92          return;
93
94        /* Reenvia pela porta serial */
95        Phy.sendSerial(&g_pkt);
96      }
97
98      return;
99    }
100
101   /**
102      Trata o estouro do buffer de recepcao.
103   */
104   void IntBufferOverflow()
105   {
106     /* Se for a primeira vez que a interrupcao e executada */
107     if (int_buff == 0) {
108       int_buff = 1;
109       return;
110     }
111
112     /* Esvazia o buffer recepcao e vai para o estado de RX */
113     cc1101.Strobe(CC1101_SFRX);
114     cc1101.Strobe(CC1101_SRX);
115
116     return;
117   }
118
119   /**
120      Trata o estouro do perÃƒÂ¯Ã‚Â¿Ã‚Â½odo do Timer1.
121   */
122   void IntTimer1()
123   {
124     /* Aqui podem ser colocadas tarefas periodicas. O periodo do Timer1 ÃƒÂ¯Ã‚Â¿Ã‚Â½
       de 1 segundo por padrÃƒÂ¯Ã‚Â¿Ã‚Â½o.
125       Mas pode ser mudado no setup() */
126
127     return;
128   }
129
```

8.3.2 Aba Headers.h

```
1   // Header.h : cabecalhos de classes
2
3   // Mais informacoes acesse www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli, Debora M.
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #ifndef HEADERS_H
24  #define HEADERS_H 1
25
26  /**
27   * Estrutura do pacote a ser transmitido e recebido. O pacote possui 52 bytes, sendo
     4 bytes de cabecalho de camada Fisica (Phy), 4 bytes de cabecalho
28   * de camada de Controle de Acesso ao Meio (MAC), 4 bytes de cabecalho de camada de
     Rede (Net), 4 bytes de cabecalho de camada de Transporte (Transp) e
29   * e 4 bytes de cabecalho de camada de AplicaÃƒÆ'Â‚Ã‚Æ'Â‚Ã‚£o (App). Os 36 bytes
     restantes sao reservados para payload de AD e IO.
30   */
31  typedef struct
32  {
33    /* cabecalho da camada Fisica (Phy) */
34    byte PhyHdr[4];
35
36    /* cabecalho da camada de Controle de Acesso ao Meio (MAC) */
37    byte MACHdr[4];
38
39    /* cabecalho da camada de Rede (Net) */
40    byte NetHdr[4];
41
42    /* cabecalho da camada de Transporte (Transp) */
43    byte TranspHdr[4];
44
45    /* Bytes o payload de AD */
46    byte AD0[3];
47    byte AD1[3];
48    byte AD2[3];
49    byte AD3[3];
50    byte AD4[3];
51    byte AD5[3];
52
53    /* Bytes do payload de IO */
54    byte IO0[3];
55    byte IO1[3];
56    byte IO2[3];
57    byte IO3[3];
58    byte IO4[3];
```

```
59      byte IO5[3];
60
61    } packet;
62
63    packet g_pkt;
64
65    /**
66     * Classe de camada de Rede
67     */
68    class NET
69    {
70      public:
71        NET(void);
72        inline void initialize(void);
73        inline void send(packet * pkt);
74        inline void receive(packet * pkt);
75
76        byte my_addr;
77      private:
78
79    };
80
81    /* Objeto de acesso a classe da camada de Rede */
82    extern NET Net;
83
84    /**
85     * Classe de camada de Controle de Acesso ao Meio
86     */
87    class MAC
88    {
89      public:
90        MAC(void);
91        inline void initialize(void);
92        inline void send(packet * pkt);
93        inline void receive(packet * pkt);
94
95      private:
96
97    };
98
99    /* Objeto de acesso a classe da camada de Controle de Acesso ao Meio */
100   extern MAC Mac;
101
102   /**
103    * Classe de camada Fisica
104    */
105   #define BUFFLEN  CC1101_PACKT_LEN
106   byte serialData[BUFFLEN + 1];
107
108   // para fazer led piscar
109   //#ifndef _PHY_H
110   //#define _PHY_H 1
111   #define SET_RX_LED(state)   if(RxLedBlink) digitalWrite(6, state);
112   #define SET_TX_LED(state)   if(TxLedBlink) digitalWrite(4, state);
113
114   class PHY
115   {
116     public:
117       PHY(void);
118       inline void initialize();
119       inline void send(packet * pkt);
120       inline int receive(packet * pkt);
121
122       void sendSerial(packet * pkt);
123       void receiveSerial(void);
124
125       byte txFifoFree(void);
126       void setChannel(byte channel);
127       void setPower(byte power);
128       void setFreqOffset(byte freq_offset);
129
130       boolean carrierSense(void);
131
```

```
132       byte power;                 /* Potencia */
133       byte channel;               /* Canal */
134       byte freq_offset;           /* Offset de frequencia */
135       int serial_baudrate;        /* Serial baudrate */
136
137       // adiciona led piscar
138       bool RxLedBlink;
139       bool TxLedBlink;
140
141     private:
142       int initCC1101Config(void);
143       void configWatchdog(int time);
144       void strobe_idle_wait(void);
145   };
146
147   /* Objeto de acesso a classe da camada Fisica */
148   extern PHY Phy;
149
150   /* Configuracao de registradores do CC1101. Obtidos atraves do SmartRF Studio 7 */
151   // Address Config = No address check
152   // Base Frequency = 915.000000
153   // CRC Autoflush = false
154   // CRC Enable = true
155   // Carrier Frequency = 915.000000
156   // Channel Number = 0
157   // Channel Spacing = 124.969482
158   // Data Format = Normal mode
159   // Data Rate = 4.79794
160   // Deviation = 14.282227
161   // Device Address = 0
162   // Manchester Enable = false
163   // Modulated = true
164   // Modulation Format = 2-FSK
165   // PA Ramping = false
166   // Packet Length = 52
167   // Packet Length Mode = Fixed packet length mode. Length configured in PKTLEN
      register
168   // Preamble Count = 4
169   // RX Filter BW = 58.035714
170   // Sync Word Qualifier Mode = 30/32 sync word bits detected
171   // TX Power = 0
172   // Whitening = false
173
174   const byte CC1101_registerSettings[CC1101_NR_OF_CONFIGS][CC1101_NR_OF_REGISTERS]
      PROGMEM = {
175   {
176      0x04,  // IOCFG2  GDO2 Output Pin Configuration
177     0x07,  // IOCFG0  GDO0 Output Pin Configuration
178     0x47,  // FIFOTHR  RX FIFO and TX FIFO Thresholds
179     0x34,  // PKTLEN  Packet Length
180     0x04,  // PKTCTRL1  Packet Automation Control
181     0x04,  // PKTCTRL0  Packet Automation Control
182     0x00,  // ADDR  Device Address
183     0x00,  // CHANNR  Channel Number
184     0x06,  // FSCTRL1  Frequency Synthesizer Control
185     0x00,  // FSCTRL0  Frequency Synthesizer Control
186     0x23,  // FREQ2  Frequency Control Word, High Byte
187     0x31,  // FREQ1  Frequency Control Word, Middle Byte
188     0x3B,  // FREQ0  Frequency Control Word, Low Byte
189     0xF7,  // MDMCFG4  Modem Configuration
190     0x83,  // MDMCFG3  Modem Configuration
191     0x03,  // MDMCFG2  Modem Configuration
192     0xA2,  // MDMCFG1  Modem Configuration
193     0x3B,  // MDMCFG0  Modem Configuration
194     0x31,  // DEVIATN  Modem Deviation Setting
195     0x18,  // MCSM0  Main Radio Control State Machine Configuration
196     0x16,  // FOCCFG  Frequency Offset Compensation Configuration
197     0x6C,  // BSCFG  Bit Synchronization Configuration
198     0x43,  // AGCCTRL2  AGC Control
199     0x40,  // AGCCTRL1  AGC Control
200     0x91,  // AGCCTRL0  AGC Control
201     0x56,  // FREND1  Front End RX Configuration
202     0x10,  // FREND0  Front End TX Configuration
```

```
203     0xE9,  // FSCAL3  Frequency Synthesizer Calibration
204     0x2A,  // FSCAL2  Frequency Synthesizer Calibration
205     0x00,  // FSCAL1  Frequency Synthesizer Calibration
206     0x1F,  // FSCAL0  Frequency Synthesizer Calibration
207     0x59,  // FSTEST  Frequency Synthesizer Calibration Control
208     0x81,  // TEST2  Various Test Settings
209     0x35,  // TEST1  Various Test Settings
210     0x09,  // TEST0  Various Test Settings
211   }
212   };
213
214   const byte CC1101_paTable[CC1101_NR_OF_CONFIGS][CC1101_PA_TABLESIZE] PROGMEM ={
215   // -30  -20   -15  -10    0   5    7   10
216     {0x03,0x0E,0x1E,0x27,0x8E,0x84,0xCC,0xC3},    // Configuracao 0
217   };
218
219   #endif
220
```

8.3.3 ABA _1_Phy

```
1   // Phy : classe da camada Fisica
2
3   // Mais informacoes acesse www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli, Debora M.
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /** Construtor da camada Fisica. */
26
27  PHY::PHY()
28  {
29   /********** AJUSTE DE Potencia ************/
30   /* A Potencia de transmissao pode ser escolhida entre 8 valores
     possÃƒÂ¯Ã‚Â¿Ã‚Â½veis (0 a 7).
31    * Em geral a Potencia e ajustada para o maximo de 7 (10 dBm).
32    * Abaixo esta a tabela que relaciona o numero com a Potencia de transmissao.
33    * | -30 | -20 | -15 | -10 |   0 |   5 |   7 |  10 |   - Potencia em dBm
34    * |   0 |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   - numero que deve ser
     colocado na Potencia
35    * !! ATENÃƒÂ¯Ã‚Â¿Ã‚Â½ÃƒÂ¯Ã‚Â¿Ã‚Â½O !!: Potencias maiores que 0dBm podem fazer com
     que as placas percam a comunicaÃƒÂ¯Ã‚Â¿Ã‚Â½ÃƒÂ¯Ã‚Â¿Ã‚Â½o quando estÃƒÂ¯Ã‚Â¿Ã‚Â½o
     muito prÃƒÂ¯Ã‚Â¿Ã‚Â½ximas.
36    *                Recomendamos posicionÃƒÂ¯Ã‚Â¿Ã‚Â½-las com uma
     distÃƒÂ¯Ã‚Â¿Ã‚Â½ncia mÃƒÂ¯Ã‚Â¿Ã‚Â½nima de 1 metro. */
37    power = 7;
38
39    /********** CANAL DE COMUNICAÃƒÂ¯Ã‚Â¿Ã‚Â½ÃƒÂ¯Ã‚Â¿Ã‚Â½O **********/
40    /* Pode ser escolhido de 0 a 65. Os canais estÃƒÂ¯Ã‚Â¿Ã‚Â½o espaÃƒÂ¯Ã‚Â¿Ã‚Â½ados
     de 200kHz e dispostos na faixa de 915 a 928 MHz */
41    channel = 3;                /* Canal */
42
43    /********** AJUSTE DE OFFSET **********/
44    /* A diferenÃƒÂ¯Ã‚Â¿Ã‚Â½a de frequÃƒÂ¯Ã‚Â¿Ã‚Â½ncia dos cristais de cada placa deve
     ser compensada.
45     * Este valor esta escrito em cada placa do BE900. */
46    freq_offset = 0x30;         /* Offset de Frequencia */
47
48    /********** VELOCIDADE DA SERIAL **********/
49    serial_baudrate = 9600;     /* Serial baudrate */
50  }
51
52  /** Inicializa a camada FÃƒÂ¯Ã‚Â¿Ã‚Â½sica. */
53
54  void PHY::initialize(void)
```

```
55  {
56    /* Configurando o serial baudrate */
57    Serial.begin(serial_baudrate);
58
59    //* Inicializa o transceptor CC1101 */
60    cc1101.PowerOnStartUp();
61
62    /* Inicializa a Configuracao do transceptor CC1101 */
63    initCC1101Config();
64
65    /* Configura o canal a ser usada */
66    setChannel(channel);
67
68    /* Configura a Potencia a ser usada */
69    setPower(power);
70
71    /* Configura o offset de frequÃƒÂ¯Ã‚Â¿Ã‚Â½ncia */
72    setFreqOffset(freq_offset);
73
74    /************************
75     * CONTROLE DOS LEDs    *
76     ************************/
77    /* O Led VERDE (Pino 6) pisca ao RECEBER um pacote, mesmo que ele nao seja
      processado.
78     * Para desabilitar essa funcao, escreva 0 na variavel abaixo. */
79    Phy.RxLedBlink = 1;
80
81    /* O Led VERMELHO (Pino 4) pisca ao ENVIAR um pacote.
82     * Para habilitar essa funcao, escreva 1 na variavel abaixo. */
83    Phy.TxLedBlink = 0;
84
85    /* Acende o Led vermelho nos kits DK101, DK102 e DK103 indicando que o sensor foi
      acionado */
86    digitalWrite (4, HIGH);
87
88  }
89
90  /** Recebe dados da porta serial. */
91
92  void PHY::receiveSerial(void)
93  {
94    byte len;             /* Tamanho do dado recebido pela porta serial */
95    byte fifoSize = 0;    /* Tamanho do espaco livre no FIFO de TX */
96
97    static byte pos = 0;  /* Total de bytes recebidos pela porta serial */
98
99    /* Le a porta serial e incrementa o total de bytes recebidos */
100   len = Serial.available() + pos;
101
102   /* Processa no maximo BUFFLEN bytes */
103   if (len > BUFFLEN )
104   {
105     len = BUFFLEN;
106   }
107
108   /* Verifica quanto espaco temos no FIFO de TX */
109   fifoSize = Phy.txFifoFree();   /* O fifoSize deve ter o numero de bytes atualmente
      livre no FIFO de TX */
110
111   /* Reinicializa as variaveis e sai da funcao */
112   if ( fifoSize <= 0)
113   {
114     Serial.flush();
115     pos = 0;
116     return;
117   }
118
119   /* Evita estourar o FIFO de TX */
120   if (len > fifoSize)
121   {
122     len = fifoSize;
123   }
124
```

```
125     /* Finalmente escreve os dados lidos da serial no FIFO de TX */
126     for (byte i = pos; i < len; i++)
127     {
128       serialData[i] = Serial.read();  /* serialData eh o nosso buffer global */
129     }
130
131     /* Atraso de 1 milissegundo */
132     delayMicroseconds(1000);
133
134     /* Verifica se existem mais dados para receber */
135     if ((Serial.available() > 0)  && (len < CC1101_PACKT_LEN))
136     {
137       pos = len;  /* Mantem a quantidade de bytes ja recebidos e espera pela proxima
        entrada nessa funcao */
138       return;
139     }
140
141     if (len == sizeof(packet))
142     {
143       /* Aguarda enquanto o canal esta sendo utilizado */
144       while(Phy.carrierSense());
145
146       /* Envia a mensagem recebida pelo RF */
147       Phy.send((packet *)serialData);
148
149       /* O buffer da serial esta livre novamente */
150       pos = 0;
151     }
152     else
153     {
154       Serial.flush();
155       pos = 0;
156       return;
157     }
158   }
159
160   /**
161    * Transmite dados pela porta serial.
162    */
163   void PHY::sendSerial(packet * pkt)
164   {
165     /* Escreve o pacote inteiro na porta serial */
166     Serial.write((byte *)pkt, sizeof(packet));
167   }
168
169   /**
170    * Le o espaco disponivel no FIFO de TX.
171    */
172   byte PHY::txFifoFree(void)
173   {
174     byte size;
175
176     cc1101.Read(CC1101_TXBYTES, &size);
177
178     /* Trata um possivel underflow to FIFO de TX */
179     if (size >= 64)
180     {
181       cc1101.Strobe(CC1101_SFTX);
182       cc1101.Read(CC1101_TXBYTES,&size);
183     }
184
185     return (CC1101_FIFO_SIZE - size);
186   }
187
188   /**
189    * Ajusta o canal a ser usado.
190    */
191   void PHY::setChannel(byte channel)
192   {
193     cc1101.Write(CC1101_CHANNR, channel);
194   }
195
196   /**
```

```
197    * Ajusta a Potencia a ser usada.
198    */
199   void PHY::setPower(byte power)
200   {
201     cc1101.setPA(0, power);
202   }
203
204   /**
205    * Ajusta o offset de frequencia.
206    */
207   void PHY::setFreqOffset(byte freq_offset)
208   {
209     cc1101.Write(CC1101_FSCTRL0, freq_offset);
210   }
211
212   /**
213    * Inicializa a Configuracao do CC1101.
214    */
215   int PHY::initCC1101Config(void)
216   {
217     /* Carrega a primeira Configuracao (pode ser inserida mais de uma Configuracao no
       cÃƒÂ¯Ã‚Â¿Ã‚Â½digo) */
218     cc1101.Setup(0);
219
220     /* Configura o endereco do rÃƒÂ¯Ã‚Â¿Ã‚Â½dio */
221     cc1101.Write(CC1101_ADDR, Net.my_addr);
222
223     /* Configura a Potencia para a maxima possivel (7) */
224     cc1101.setPA(0, 7);
225
226     /* Coloca o CC1101 no estado de RX */
227     cc1101.Strobe(CC1101_SIDLE);
228     delay(1);
229     cc1101.Write(CC1101_MCSM1 ,   0x0F );
230     cc1101.Strobe(CC1101_SFTX);
231     cc1101.Strobe(CC1101_SFRX);
232     cc1101.Strobe(CC1101_SRX);
233
234     return OK;
235   }
236
237   /**
238    * Retorna o status de presenca de portadora no canal (Carrier Sense)
239    * return   1 se o canal esta ocupado, 0 se o canal esta livre
240    */
241   boolean PHY::carrierSense(void)
242   {
243     byte cs;
244
245     /* O status da portadora e lido no registrador PKTSTATUS */
246     cc1101.Read(CC1101_PKTSTATUS,&cs);
247     /* O bit de Carrier Sense e o bit 6 */
248     cs &= 0x40;
249
250     if (cs)
251       return true;
252     else
253       return false;
254
255   }
256
257   /**
258    * Configura o valor do timeout do WatchDog
259    * 0=16ms, 1=32ms, 2=64ms, 3=128ms, 4=250ms, 5=500ms, 6=1sec, 7=2sec, 8=4sec, 9=8sec
260    */
261   void PHY::configWatchdog(int time) {
262
263     byte value;
264
265     if (time > 9 ) time = 9;
266     value = time & 7;
267     if (time > 7) value |= (1<<5);
268     value |= (1<<WDCE);
```

```
269
270     /* Habilita a interrupcao de WatchDog no Satus Register */
271     MCUSR &= ~(1<<WDRF);
272
273     /* Configura as flags do registrador de WatchDog */
274     WDTCSR |= (1<<WDCE) | (1<<WDE);
275
276     /* Configura o valor do timeout */
277     WDTCSR = value;
278
279     /* Habilita a interrupcao do WatchDog */
280     WDTCSR |= _BV(WDIE);
281
282   }
283
284   /**
285    *    Strobe a command in order to the CC1101 go to the IDLE state
286    */
287
288   void PHY::strobe_idle_wait( void )
289   {
290     byte current_state;
291
292     /* Strobe to IDLE state */
293     cc1101.Strobe(CC1101_SIDLE);
294
295     do
296     {
297       /* Read current state */
298       cc1101.Read( CC1101_MARCSTATE, &current_state);
299
300     /* Until in IDLE state */
301     } while ( current_state != 0x01 );
302   }
303
304   /**
305    * Envia dados pelo RF.
306    */
307   inline void PHY::send(packet * pkt)
308   {
309     byte current_state;
310     byte *txData = (byte *)pkt;
311
312     /* Coloca o CC1101 no estado IDLE */
313     cc1101.Strobe(CC1101_SIDLE);
314
315     /* Escreve uma rajada com os dados a serem transmitidos */
316     cc1101.WriteBurst(CC1101_TXFIFO, txData, sizeof(packet));
317
318     /* Vai para o estado TX */
319     cc1101.Strobe(CC1101_STX);
320
321     /* Aguarda enquanto todos os bytes estÃƒÂ¯Ã‚Â¿Ã‚Â½o sendo transmitidos */
322     while(1)
323     {
324       byte size;
325           cc1101.Read(CC1101_TXBYTES, &size);
326       if( size == 0 )
327       {
328         break;
329       }
330       else
331       {
332         cc1101.Strobe(CC1101_STX);
333       }
334     }
335
336     /* Espera que o modulo retorne para o estado de RX */
337     do
338     {
339       cc1101.Read(CC1101_MARCSTATE, &current_state);
340     } while ( current_state != 0x0D );
341   }
```

8.3.4 Aba _2_MAC

```
1   // MAC : classe da camada de Controle de Acesso ao Meio
2
3   // Mais informacoes acesse www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli, Debora M.
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada de Controle de Acesso ao Meio.
27   */
28  MAC::MAC()
29  {
30  }
31
32  /**
33   * Inicializa a camada de Controle de Acesso ao Meio.
34   */
35  void MAC::initialize(void)
36  {
37  }
38
39  /**
40   * Envia o pacote para a camada inferior
41   */
42  inline void MAC::send(packet * pkt)
43  {
44    return;
45  }
46
47  /**
48   * Recebe o pacote da camada inferior
49   */
50  inline void MAC::receive(packet * pkt)
51  {
52    return;
53  }
54
55  /* InstanciaÃƒÆ'Â‚Â§Ã£o do objeto de acesso Ã£Æ'Â‚Â  classe da camada de
    Controle de Acesso ao Meio */
56  MAC Mac = MAC();
57
```

Aba _3_Net

```
1   // NET : classe da camada de Rede
2
3   // Mais informacoes acesse www.radiuino.cc
4   // Copyright (c) 2015
5   // Author: Pedro Henrique Gomes, Omar C. Branquinho, Tiago T. Ganselli, Debora M.
    Ferreira
6   // Versao 2.2: 20/01/2015
7
8   // Este arquivo e parte da plataforma Radiuino
9   // Este programa e um software livre; voce pode redistribui-lo e/ou modifica-lo
    dentro dos termos da Licenca Publica Geral Menor GNU
10  // como publicada pela Fundacao do Software Livre (FSF); na versao 2 da Licenca, ou
    (na sua opniao) qualquer futura versao.
11  // Este programa e distribuido na esperanca que possa ser  util, mas SEM NENHUMA
    GARANTIA; sem uma garantia implicita
12  // de ADEQUACAO a qualquer MERCADO ou APLICACAO EM PARTICULAR. Veja a Licenca
    Publica Geral Menor GNU para maiores detalhes.
13  // Voce deve ter recebido uma copia da Licenca Publica Geral Menor GNU junto com
    este programa, se nao, escreva para a Fundacao
14  // do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
    USA
15
16  // This library is free software; you can redistribute it and/or modify it under the
    terms of the GNU Lesser General Public License
17  // as published by the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version. This library
18  // is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY
19  // or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
    for more details. You should have received a copy
20  // of the GNU Lesser General Public License along with this library; if not, write
    to the Free Software Foundation, Inc., 51 Franklin St,
21  // Fifth Floor, Boston, MA  02110-1301  USA
22
23  #include "Headers.h"
24
25  /**
26   * Construtor da camada de Rede.
27   */
28  NET::NET()
29  {
30    my_addr = 0;    /* EndereÃƒÆ'Ã‚Â§o */
31  }
32
33  /**
34   * Inicializa a camada de Controle de Acesso ao Meio.
35   */
36  void NET::initialize(void)
37  {
38  }
39
40  /**
41   * Envia o pacote para a camada inferior
42   */
43  inline void NET::send(packet * pkt)
44  {
45    return;
46  }
47
48  /**
49   * Recebe o pacote da camada inferior
50   */
51  inline void NET::receive(packet * pkt)
52  {
53    return;
54  }
55
56  /* Instanciacao do objeto de acesso a classe da camada de Rede */
57  NET Net = NET();
```

9 ANEXO

9.1 ANEXO1 - BE900

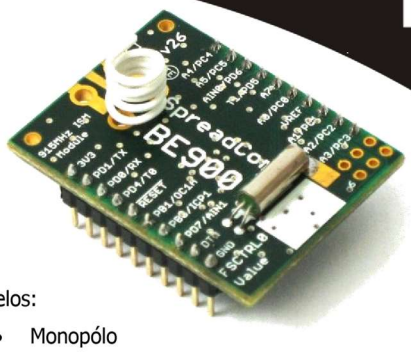
DATASHEET – ESPECIFICAÇÕES TÉCNICAS

BE900 – Módulo de comunicação sem fio

Radio it!

BE900

O MÓDULO SEM FIO. DE VERDADE.



O BE900 é um módulo com microcontrolador e transceptor integrados, que possibilita aplicações de monitoração e controle, podendo ser totalmente programado.

O BE900 é um módulo de comunicação extremamente flexível que utiliza o processador AVR Atmega328 e o transceptor TI CC1101 RF com filtro passa-faixa para maior sensibilidade e imunidade a interferência a ruído, ajustado para operar na banda não licenciada ISM de 915MHz (902 até 928 MHz).

O módulo possui modos de operação com baixo consumo de potência e um relógio de tempo real (RTC) baseado em cristal de 32kHz.

Atende à regulamentação ANATEL e FCC (para maiores informações: www.spreadcom.com.br).

Características:

- **IDE Arduino**: o BE900 pode ser programado utilizando a IDE Arduino e usufruir da maioria dos software se bibliotecas desenvolvidos para esta plataforma;
- **E/S do Arduino**: os 15 pinos de E/S disponíveis podem ser mapeados como pinos Arduino;
- **Microcontrolador**: AVR Atmega 328, microcontrolador de 8 bits com alto desempenho e baixo consumo, com 32k de ROM, 2k de RAM, 1k de EEPROM e clock de 8MHz (mais informações: www.atmel.com);
- **Comunicação sem fio**: o módulo possui um TI CC1101, transceptor de RF em um único chip para bandas não-licenciadas ISM com filtro passa-faixa para maior sensibilidade e menor interferência de ruídos, ajustado para operar na banda de 915MHz (902-907,5MHz e 915-928MHz). A placa possibilita a soldagem de um conector SMA para uso profissional;
- **Refinamento do RF**: Todos os módulos passam por um processo de calibração de fábrica para o correto ajuste do deslocamento de frequência do circuito de RF. O valor do ajuste é informado ao usuário em cada módulo e deve ser utilizado para melhorar a qualidade de comunicação;
- **Flexibilidade de programação**: Apesar de poder ser programado via Arduino, o módulo também pode ser programado utilizando a interface AVR, com a vantagem de já ter o transceptor embarcado.

ANATEL Monopólo 2066-12-8136 (01)07898949708018

ANATEL Reto 2066-12-8136 (01)07898949708025

ANATEL 90° 2066-12-8136 (01)07898949708032

Modelos:

- Monopólo
- SMA Reto
- SMA 90°

Especificações técnicas

RF	
Frequência de Operação	902-907,5MHz e 915- 928MHz
Modulação	2FSK (Configurável)
Tecnologia	Modulação Digital
Taxa de dados do RF	Até 250kbps
Potência de TX	Até +10dBm
Sensibilidade RX	Até -112dBm (com ~1% de PER)
Alcance Indoor	Até 100m
Alcance Outdoor	Até 500m
Regulamentação	FCC, Anatel, Australia

Placa	
Dimensões	24,4mm x 32mm x 10,5mm
Número de Pinos	20
Espaçamento dos pinos	2mm
Conector de RF (Opcional)	SMA Reto ou SMA 90°
Pinos de E/S	E/S digitais, UART, I2C, SPI, ADC, PWM

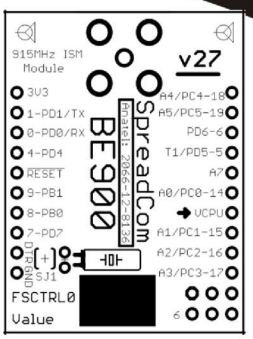
Microcontrolador	
Memória	32kB Flash, 2kB RAM, 1kB EEPROM
Clock da CPU	8MHz
RTC (Relógio de tempo real)	32768kHz (+/- 10ppm)
Conversores AD	Até 7 canais de 10 bits
Saídas PWM	2
Entradas/Saídas Digitais	Até 14 entradas ou saídas

"Este equipamento opera em caráter secundário, isto é, não tem direito à proteção contra interferência prejudicial, mesmo de estações do mesmo tipo, e não pode causar interferência a sistemas operando em caráter primário."

Radioit Eletrônica LTDA
Campinas-SP
Tel: +55 19 32012489
www.radioit.com.br

Data: 26/11/2012 – Versão: 1.5

Especificações Elétricas	Min.	Tip.	Max	Un.
Tensão de Entrada	3.0	3.3	3.6	V_{DC}
Corrente de transmissão		34.5		mA
Corrente de recepção		18.1		mA
Corrente de inatividade		5.2		mA
Corrente de dormência		<0.3		mA
Temperatura de operação	-50		125	°C

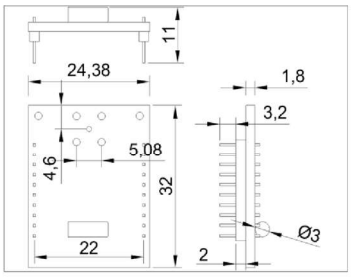


Pinagem ISP (*In-System Programming*):

Pino	Função	Tipo	Função no ATmega328	Pino do Arduino
1	MISO	Entrada/Saída	MISO (SPI Bus Master Input/Slave Output), PCINT4	12 (DIO)
2	SCK	Entrada/Saída	SCK (SPI Bus Master clock Input), PCINT5	13 (DIO)
3	RESET	Entrada	Pino de Reset	-
4	GND	Terra	Terra	-
5	MOSI	Entrada/Saída	MOSI (SPI Bus Master Output/Slave Input), OC2A (Saída da Comparação A do Timer/Counter2), PCINT3	11 (DIO)
6	3V3	Entrada alim.	VCC (3.3V)	-

Notas: 1)PCINTxx (Interrupção xx do Pino)

Dimensões (em mm):



Pinagem do módulo:

Pino	Nome	Tipo	Função no ATmega328	Pino do Arduino
1	3V3	Entrada alim.	VCC (3.3V)	-
2	PD1/TX	Entrada/Saída	TXD (Pino de saída da USART), PCINT17	1(E/S digital)
3	PD0/RX	Entrada/Saída	RXD (Pino de entrada da USART), PCINT16	0(E/S digital)
4	PD4/T0	Entrada/Saída	XCK (Clock externo da USART), T0 (Entrada externa do Timer/Counter 0), PCINT20	4(E/S digital)
5	/RESET	Entrada	Pino de reset	-
6	PB1/OC1A	Entrada/Saída	OC1A (Saída da Comparação A do Timer/Counter1), PCINT1	9(E/S digital)
7	PB0/ICP1	Entrada/Saída	ICP1 (Entrada de captura do Timer/Counter1), CLKO (Saída do clock do sistema), PCINT0	8(E/S digital)
8	PD7/AIN1	Entrada/Saída	AIN1 (Entrada negativa do comparador analógico), PCINT23	7(E/S digital)
9	DTR	Entrada	Para a programação do ATmega328	-
10	GND	Terra	Terra	-
11	A3/PC3	Entrada/Saída	ADC3, PCINT11	3(Entrada Analógica)/ 17(E/S digital)
12	A2/PC2	Entrada/Saída	ADC2, PCINT10	2(Entrada Analógica)/ 16(E/S digital)
13	A1/PC1	Entrada/Saída	ADC1, PCINT9	1(Entrada Analógica)/ 15(E/S digital)
14	VREF	Entrada/Saída	Referência da voltagem do ADC(AREF)	-
15	A0/PC0	Entrada/Saída	ADC0, PCINT8	0(Entrada Analógica)/ 14(E/S digital)
16	A7	Entrada	ADC7	7(Entrada Analógica)
17	T1/PD5	Entrada/Saída	T1 (Entrada externa do Timer/Counter 1), OC0B (Saída da Comparação B do Timer/Counter0), PCINT21	5(E/S digital)
18	AIN0/PD6	Entrada/Saída	AIN0 (Entrada positiva do comparador analógico), OC0A (Saída da Comparação A do Timer/Counter0), PCINT22	6(E/S digital)
19	A5/PC5	Entrada/Saída	ADC5 (Entrada do ADC canal 5), SCL (linha de clock do barramente serial), PCINT13	5(Entrada Analógica)/ 19(E/S digital)
20	A4/PC4	Entrada/Saída	ADC4 (Entrada do ADC canal 4), SDA (linha de dados do barramento serial), PCINT12	4(Entrada Analógica)/ 18(E/S digital)

Notas: 1) PCINTxx (Interrupção xx do Pino); 2)ADCx (Canal x de entrada do ADC)

9.2 ANEXO 2 – CN3063

CONSONANCE

Lithium Ion Battery Charger for Solar-Powered Systems

CN3063

General Description:

The CN3063 is a complete constant-current /constant voltage linear charger for single cell Li-ion and Li Polymer rechargeable batteries. The device contains an on-chip power MOSFET and eliminates the need for the external sense resistor and blocking diode. An on-chip 8-bit ADC can adjust charging current automatically based on the output capability of input power supply, so CN3063 is ideally suited for solar powered system. Furthermore, the CN3063 is specifically designed to work within USB power specifications. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The regulation voltage is internally fixed at 4.2V with 1% accuracy, it can also be adjusted with an external resistor. The charge current can be programmed externally with a single resistor. When the input supply is removed, the CN3063 automatically enters a low power sleep mode , dropping the battery drain current to less than 3uA. Other features include undervoltage lockout, automatic recharge, battery temperature sensing and charging/termination indicator.
The CN3063 is available in a thermally enhanced 8-pin SOP package.

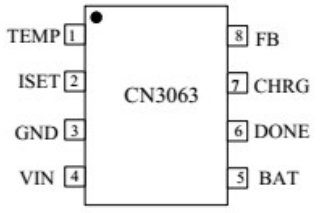
Applications:

- Solar Powered System
- Digital Still Cameras
- MP3 Players
- Bluetooth Applications
- Portable Devices
- Chargers

Features:

- On-chip 8-bit ADC can adjust charging current automatically based on the output capability of input power supply
- Suitable for Solar-Powered System
- On-chip Power MOSFET
- No external Blocking Diode or Current Sense Resistors Required
- Preset 4.2V Regulation Voltage with 1% Accuracy, adjustable with an external resistor
- Precharge Conditioning for Reviving Deeply Discharged Cells and Minimizing Heat Dissipation During Initial Stage of Charge
- Continuous Programmable Charge Current Up to 600mA
- Constant-Current/Constant-Voltage Operation with Thermal Regulation to Maximize Charge Rate Without Risk of Overheating
- Automatic Low-Power Sleep Mode When Input Supply Voltage is Removed
- Status Indication for LEDs or uP Interface
- C/10 Charge Termination
- Automatic Recharge
- Battery Temperature Sensing
- Available in SOP8 Package
- Pb-free Available

Pin Assignment



Typical Application Circuit

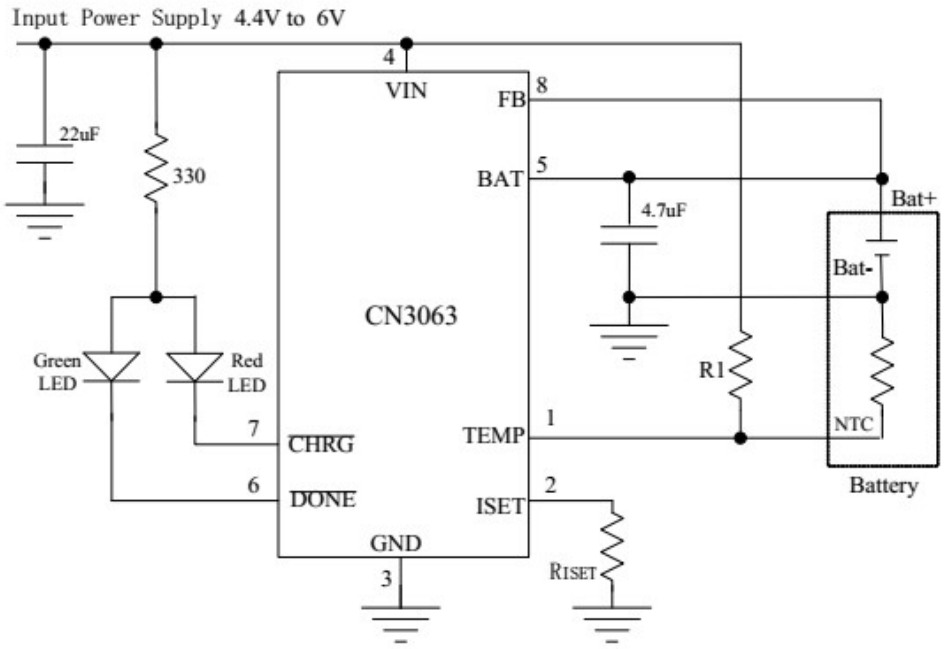


Figure 1 Typical Application Circuit(Constant Voltage Level 4.2V)

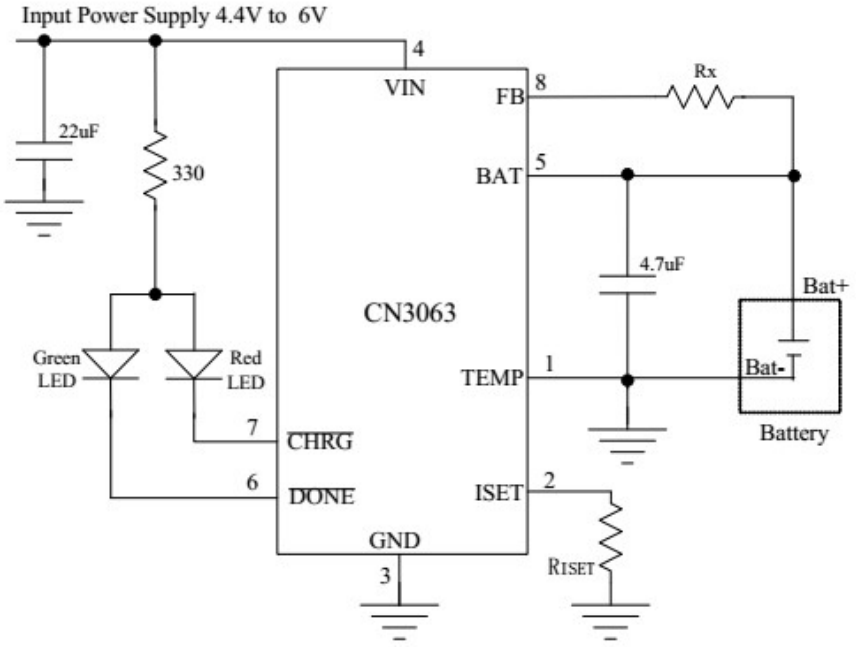


Figure 2 Application Circuit(Adjust Constant Voltage Level with Rx)

In Figure 2, the BAT pin's voltage in constant voltage mode is given by the following equation:

$$Vbat = 4.2 + 3.04 \times 10^{-6} \times Rx$$

Where, Vbat is in volt

Rx is in ohm

Block Diagram

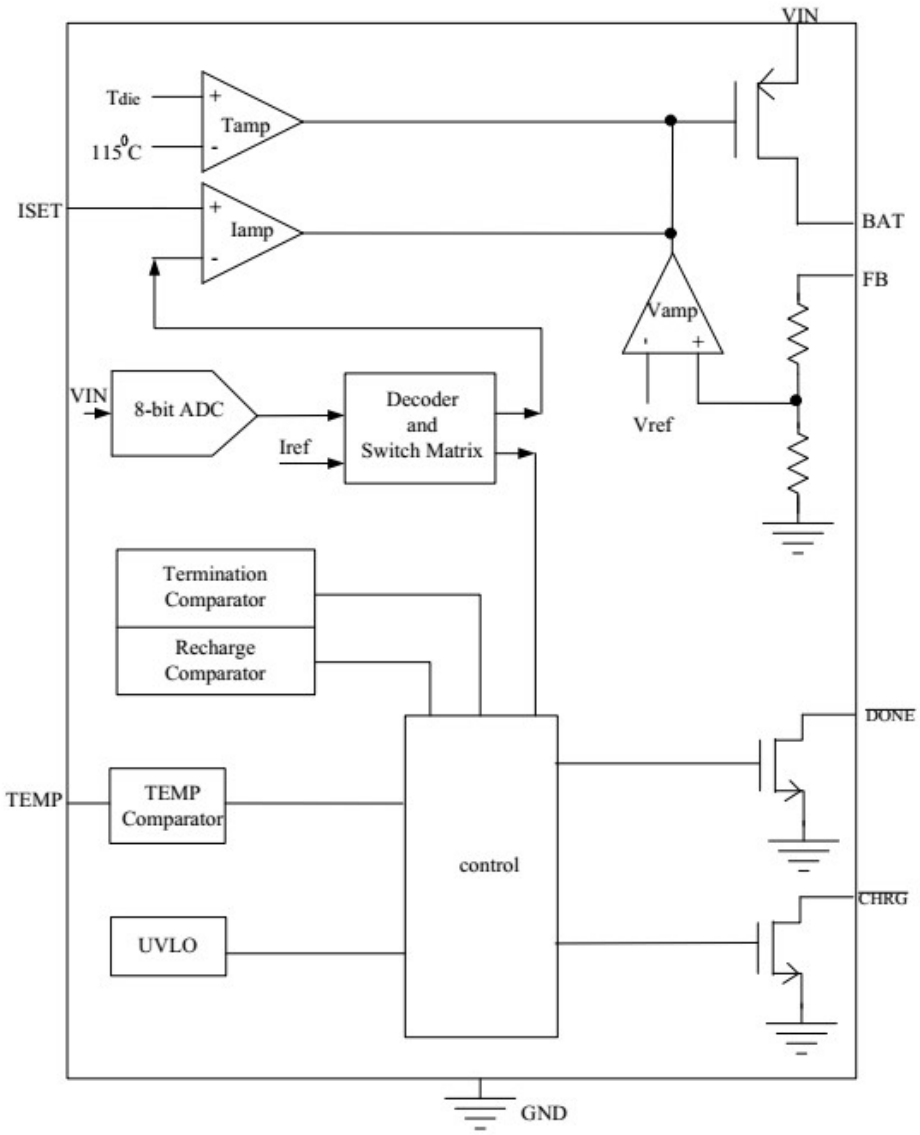


Figure 3 Block Diagram

Pin Description

Pin No.	Name	Function Description
1	TEMP	**Temperature Sense Input.** Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 46% of input supply voltage V_{IN} for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. If TEMP's voltage level is above 46% of input supply voltage for more than 0.15S, battery fault state is released, and charging will resume. The temperature sense function can be disabled by grounding the TEMP pin.
2	ISET	**Constant Charge Current Setting and Charge Current Monitor Pin.** The charge current is set by connecting a resistor R_{ISET} from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V. In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows: $I_{CH} = (V_{ISET} / R_{ISET}) \times 900$
3	GND	**Ground Terminal.**
4	VIN	**Positive Input Supply Voltage.** V_{IN} is the power supply to the internal circuit. When V_{IN} drops to within 40mv of the BAT pin voltage, CN3063 enters low power sleep mode, dropping BAT pin's current to less than 3uA.
5	BAT	**Battery Connection Pin.** Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 3uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.
6	$\overline{DONE}$	**Open-Drain Charge termination Status Output.** In charge termination status, $\overline{DONE}$ is pulled low by an internal switch; Otherwise $\overline{DONE}$ pin is in high impedance state.
7	$\overline{CHRG}$	**Open Drain Charge Status Output.** When the battery is being charged, the $\overline{CHRG}$ pin is pulled low by an internal switch, otherwise $\overline{CHRG}$ pin is in high impedance state.
8	FB	**Battery Voltage Kevin Sense Input.** This Pin can Kelvin sense the battery voltage; Also the regulation voltage in constant voltage mode can be adjusted by connecting an external resistor between FB pin and BAT pin.

Absolute Maximum Ratings

All Terminal Voltage……－0.3V to 6.5V
BAT Short-Circuit Duration……...Continuous
Storage Temperature…..…….－65℃ to 150℃
Lead Temperature(Soldering)…….....300℃
Maximum Junction Temperature…...150℃
Operating Temperature…...－40℃ to 85℃
Thermal Resistance (SOP8)……TBD

Stresses beyond those listed under 'Absolute Maximum Ratings' may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specifications is not implied. Exposure to Absolute Maximum Rating Conditions for extended periods may affect device reliability.

Electrical Characteristics

(VIN=5V, T_A=－40℃ to 85℃, Typical Values are measured at T_A=25℃,unless otherwise noted)

Parameters	Symbol	Test Conditions	Min	Typ	Max	Unit
Input Supply Voltage	VIN		4.4		6	V
Operating Current	I_{VIN}	Charge Termination Mode	400	650	950	uA
Undervoltage Lockout	Vuvlo	VIN falling		3.7	3.9	V
Undervoltage Lockout Hysteresis	Huvlo			0.1		V
Regulation Voltage	V_{REG}	Constant Voltage Mode	4.158	4.2	4.242	V
BAT pin Current	I_{BAT}	R_{ISET}=3.6K, V_{BAT}=3.6V	400	500	600	mA
		R_{ISET}=3.6K, V_{BAT}=2.4V	25	50	75	
		V_{BAT}=V_{REG}, standby mode	1.75	3.5	7	uA
		VIN=0V, sleep mode			3	
Precharge Threshold						
Precharge Threshold	V_{PRE}	Voltage at BAT pin rising	2.9	3.0	3.1	V
Precharge Threshold Hysteresis	H_{PRE}			0.1		V
Charge Termination Threshold						
Charge Termination Threshold	Vterm	Measure voltage at ISET pin	0.18	0.22	0.26	V
Recharge Threshold						
Recharge Threshold	V_{RECH}			V_{REG}－0.1		V
Sleep Mode						
Sleep Mode Threshold	V_{SLP}	V_{IN} from high to low, measures the voltage (V_{IN}－V_{BAT})		40		mv
Sleep mode Release Threshold	V_{SLPR}	V_{IN} from low to high, measures the voltage (V_{IN}－V_{BAT})		90		mv
ISET Pin						
ISET Pin Voltage	V_{ISET}	Precharge mode		0.2		V
		Constant current mode		2.0		
TEMP PIN						
Input Threshold	V_{TEMP}		43.5	46	48.5	%V_{IN}
TEMP input Current		TEMP to V_{IN} or to GND			0.5	uA
$\overline{DONE}$ **Pin**						
$\overline{DONE}$ Sink Current	I_{DONE}	V_{DONE}=0.3V, termination mode		10		mA
$\overline{DONE}$ Leakage Current		V_{DONE}=6V, charge mode			1	uA
$\overline{CHRG}$ **Pin**						
$\overline{CHRG}$ Sink Current	I_{CHRG}	V_{CHRG}=0.3V, Charge status		10		mA
$\overline{CHRG}$ Leakage Current		V_{CHRG}=6V, termination mode			1	uA

Detailed Description

The CN3063 is a linear battery charger designed primarily for charging single cell lithium-ion or lithium-polymer batteries. Featuring an internal P-channel power MOSFET, the charger uses a constant-current/constant-voltage to charge the batteries. Continuous charge current can be programmed up to 600mA with an external resistor. No blocking diode or sense resistor is required. The on-chip 8-bit ADC can adjust charging current automatically based on the output capability of input power supply, so CN3063 is ideally suited for the solar-powered systems, or the applications that need to charge lithium-ion battery or lithium polymer battery with an input power supply whose output capability is limited. The open-drain output $\overline{DONE}$ and $\overline{CHRG}$ indicates the charger's status. The internal thermal regulation circuit reduces the programmed charge current if the die temperature attempts to rise above a preset value of approximately 115℃. This feature protects the CN3063 from excessive temperature, and allows the user to push the limits of the power handling capability of a given circuit board without risk of damaging the CN3063 or the external components. Another benefit of adopting thermal regulation is that charge current can be set according to typical, not worst-case, ambient temperatures for a given application with the assurance that the charger will automatically reduce the current in worst-case conditions.

The charge cycle begins when the voltage at the V_{IN} pin rises above the UVLO level, a current set resistor is connected from the ISET pin to ground. The $\overline{CHRG}$ pin outputs a logic low to indicate that the charge cycle is ongoing. At the beginning of the charge cycle, if the voltage at FB pin is below 3V, the charger is in precharge mode to bring the cell voltage up to a safe level for charging. The charger goes into the fast charge constant-current mode once the voltage on the FB pin rises above 3V. In constant current mode, the charge current is set by R_{ISET}. When the battery approaches the regulation voltage, the charge current begins to decrease as the CN3063 enters the constant-voltage mode. When the current drops to charge termination threshold, the charge cycle is terminated, $\overline{DONE}$ is pulled low by an internal switch and $\overline{CHRG}$ pin assumes a high impedance state to indicate that the charge cycle is terminated. The charge termination threshold is 10% of the current in constant current mode. To restart the charge cycle, just remove the input voltage and reapply it. The charge cycle can also be automatically restarted if the FB pin voltage falls below the recharge threshold. The on-chip reference voltage, error amplifier and the resistor divider provide regulation voltage with 1% accuracy which can meet the requirement of lithium-ion and lithium polymer batteries. When the input voltage is not present, the charger goes into a sleep mode, dropping battery drain current to less than 3uA. This greatly reduces the current drain on the battery and increases the standby time.

The charging profile is shown in the following figure:

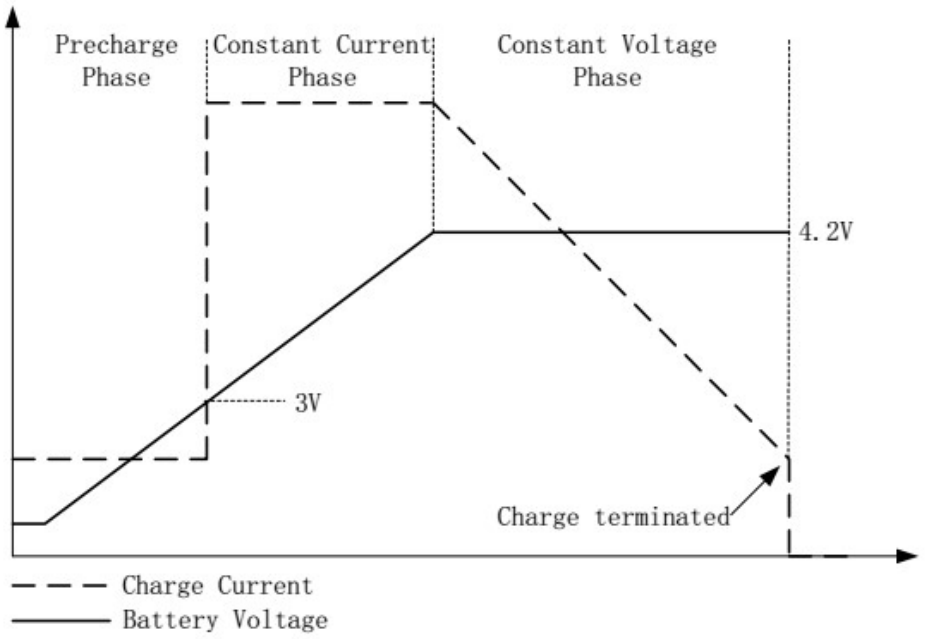


Figure 4 Charging Profile

Application Information

Undervoltage Lockout (UVLO)

An internal undervoltage lockout circuit monitors the input voltage and keeps the charger in shutdown mode until V_{IN} rises above the undervoltage lockout voltage. The UVLO circuit has a built-in hysteresis of 0.1V.

Sleep mode

There is an on-chip sleep comparator. The comparator keeps the charger in sleep mode if V_{IN} falls below sleep mode threshold(VBAT+40mv). Once in sleep mode, the charger will not come out of sleep mode until V_{IN} rises 90mv above the battery voltage.

Precharge mode

At the beginning of a charge cycle, if the battery voltage is below 3V, the charger goes into precharge mode , and the charge current is 10% of fast charge current in constant current mode.

Charging Current limited by the Output capability of Input Power Supply

If the output capability of input power supply is less than the charging current set by the resistor at ISET pin, then the on-chip 8-bit ADC will begin to function to adjust the charging current based on the output capability of input power supply. In this case, the charging current may be less than the value set by the resistor at ISET pin, but it is maximized to the output capability of input power supply on the condition that the input voltage at VIN pin is no less than 4.35V, which is the minimum operating voltage of CN3063. So the charging current can be set according to the maximum output capability of input power supply, not the worst case.

Adjusting the regulation voltage in constant voltage mode

The regulation voltage in constant voltage mode can be adjusted by an external resistor connecting between FB pin and BAT pin as shown in Figure 5:

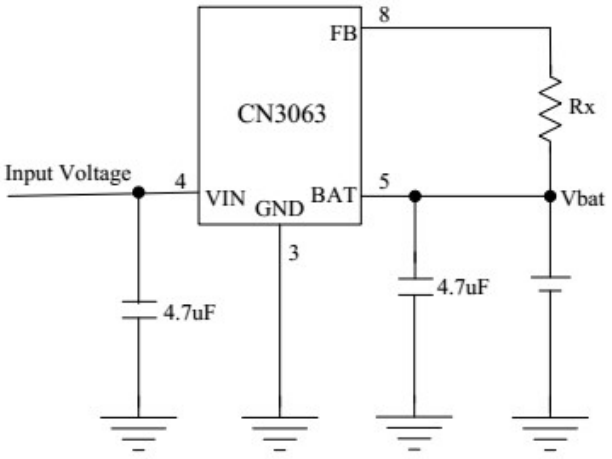


Figure 5 Adjusting Regulation Voltage in Constant Voltage Mode

In Figure 5, the regulation voltage in constant voltage mode will be given by the following equation:

$$Vbat = 4.2 + 3.04 \times 10^{-6} \times Rx$$

Where,

Vbat is in volt

Rx is in ohm

Programming Charge Current

The formula for the battery charge current in constant current mode is:

$$I_{CH} = 1800V / R_{ISET}$$

Where:

I_{CH} is the charge current in ampere

R_{ISET} is the total resistance from the ISET pin to ground in ohm

For example, if 500mA charge current is required, calculate:

$$R_{ISET} = 1800V/0.5A = 3.6k\,\Omega$$

For best stability over temperature and time, 1% metal film resistors are recommended. If the charger is in constant-temperature or constant voltage mode, the charge current can be monitored by measuring the ISET pin voltage, and the charge current is calculated as the following equation:

$$I_{CH} = (V_{ISET} / R_{ISET}) \times 900$$

Combine Two Power Inputs

Although the CN3063 allows charging from a solar power supply, a wall adapter or a USB port can also be used to charge Li-Ion/Li-polymer batteries. Figure 6 shows an example of how to combine 2 power inputs. A P-channel MOSFET, M1, is used to prevent back conducting into the 2nd power supply when the 1st power supply is present and Schottky diode, D1, is used to prevent 2nd power supply loss through the 1k Ω pull-down resistor.

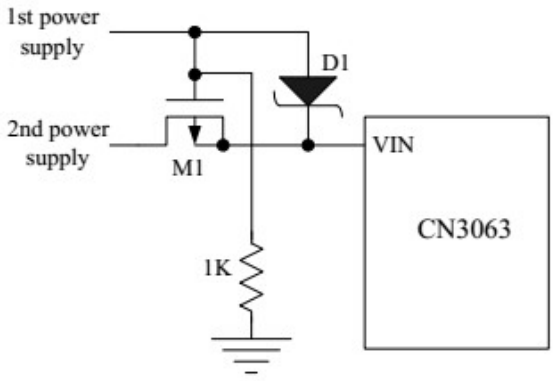


Figure 6 Combining 2 Input Power Supply

Battery Temperature Sense

To prevent the damage caused by the very high or very low temperature done to the battery pack, the CN3063 continuously senses battery pack temperature by measuring the voltage at TEMP pin.

If V_{TEMP}<(46%×VIN) for 0.15 seconds, it indicates that the battery temperature is too high or too low and the charge cycle is suspended. If V_{TEMP}>(46%×VIN) for 0.15 seconds, the charge cycle resumes.

The battery temperature sense function can be disabled by connecting TEMP pin to GND.

Recharge

After a charge cycle has terminated, if the battery voltage drops below the recharge threshold of 4.1V, a new charge cycle will begin automatically.

Constant-Current/Constant-Voltage/Constant-Temperature

The CN3063 use a unique architecture to charge a battery in a constant-current, constant-voltage, constant temperature fashion as shown in Figure 3. Amplifiers Iamp, Vamp, and Tamp are used in three separate feedback loops to force the charger into constant-current, constant-voltage, or constant-temperature mode, respectively. In constant current mode the charge current delivered to the battery equal to $1800V/R_{ISET}$. If the power dissipation of the CN3063 results in the junction temperature approaching 115℃, the amplifier Tamp will begin decreasing the charge current to limit the die temperature to approximately 115℃. As the battery voltage rises, the CN3063 either returns to constant-current mode or it enters constant voltage mode straight from constant-temperature mode.

Open-Drain Status Outputs

The CN3063 have 2 open-drain status outputs: $\overline{DONE}$ and $\overline{CHRG}$. $\overline{CHRG}$ is pulled low when the charger is in charging status, otherwise $\overline{CHRG}$ becomes high impedance. $\overline{DONE}$ is pulled low if the charger is in charge termination status, otherwise $\overline{DONE}$ becomes high impedance.

When the battery is not present, the charger charges the output capacitor to the regulation voltage quickly, then the BAT pin's voltage decays slowly to recharge threshold because of low leakage current at BAT pin, which results in a 100mv ripple waveform at BAT pin, in the meantime, $\overline{CHRG}$ pin outputs a pulse to indicate that the battery's absence. The pulse's frequency is around 10Hz when a 4.7uF output capacitor is used.

The open drain status output that is not used should be tied to ground.

V_{IN} Bypass Capacitor C_{IN}

Many types of capacitors can be used for input bypassing, C_{IN} is typically a 22uF capacitor.

For the consideration of the bypass capacitor, please refer to the Application Note AN102 from our website.

Stability

Typically a 4.7uF capacitor from BAT pin to GND is required to stabilize the feedback loop.

In constant current mode, the stability is also affected by the impedance at the ISET pin . With no additional capacitance on the ISET pin, the loop is stable with current set resistors values as high as 50KΩ. However, additional capacitance on ISET pin reduces the maximum allowed current set resistor. The pole frequency at ISET pin should be kept above 200KHz. Therefore, if ISET pin is loaded with a capacitance C, the following equation should be used to calculate the maximum resistance value for R_{ISET}:

$$R_{ISET} < 1 / (6.28 \times 2 \times 10^5 \times C)$$

In order to measure average charge current or isolate capacitive load from ISET pin, a simple RC filter can be used on ISET pin as shown in Figure 7.

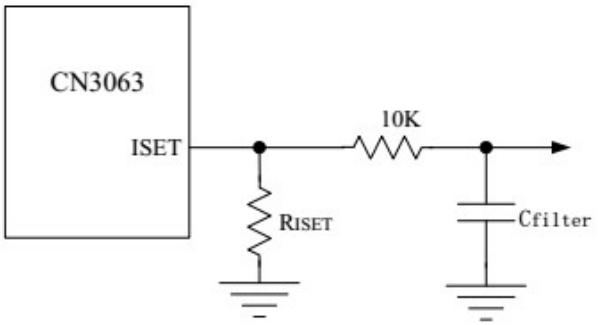


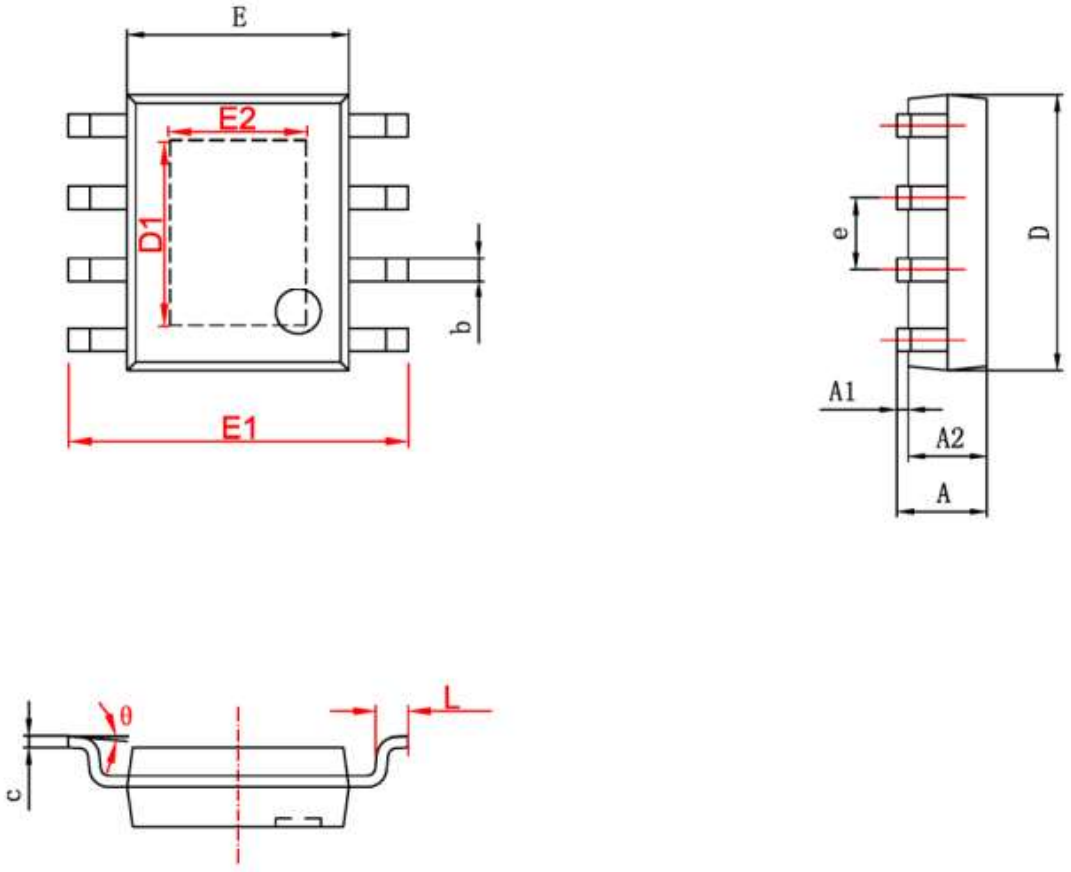
Figure 7 Isolating Capacitive Load on ISET Pin

Board Layout Considerations

1. R_{ISET} at ISET pin should be as close to CN3063 as possible, also the parasitic capacitance at ISET pin should be kept as small as possible.
2. The capacitance at VIN pin and BAT pin should be as close to CN3063 as possible.
3. During charging, CN3063's temperature may be high, the NTC thermistor should be placed far enough to CN3063 so that the thermistor can reflect the battery's temperature correctly.
4. It is very important to use a good thermal PC board layout to maximize charging current. The thermal path for the heat generated by the IC is from the die to the copper lead frame through the package lead(especially the ground lead) to the PC board copper, the PC board copper is the heat sink. The footprint copper pads should be as wide as possible and expand out to larger copper areas to spread and dissipate the heat to the surrounding ambient. Feedthrough vias to inner or backside copper layers are also useful in improving the overall thermal performance of the charger. Other heat sources on the board, not related to the charger, must also be considered when designing a PC board layout because they will affect overall temperature rise and the maximum charge current.

   The ability to deliver maximum charge current under all conditions require that the exposed metal pad on the back side of the CN3063 package be soldered to the PC board ground. Failure to make the thermal contact between the exposed pad on the backside of the package and the copper board will result in larger thermal resistance.

Package Information



字符	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A	1.350	1.750	0.053	0.069
A1	0.050	0.150	0.004	0.010
A2	1.350	1.550	0.053	0.061
b	0.330	0.510	0.013	0.020
c	0.170	0.250	0.006	0.010
D	4.700	5.100	0.185	0.200
D1	3.202	3.402	0.126	0.134
E	3.800	4.000	0.150	0.157
E1	5.800	6.200	0.228	0.244
E2	2.313	2.513	0.091	0.099
e	1.270 (BSC)		0.050 (BSC)	
L	0.400	1.270	0.016	0.050
θ	0°	8°	0°	8°