UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

INTERNET DAS COISAS APLICADA À DOMÓTICA

Autor: Fernando Mendonça de Almeida

Monografia de Conclusão do Curso de Engenharia de Computação

Orientador: Marco Túlio Chella

SÃO CRISTÓVÃO
2013

Fernando Mendonça de Almeida

INTERNET DAS COISAS APLICADA À DOMÓTICA

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Marco Túlio Chella

SÃO CRISTÓVÃO

2013

FERNANDO MENDONÇA DE ALMEIDA

INTERNET DAS COISAS APLICADA À DOMÓTICA

Trabalho de Conclusão de Curso submetido ao corpo docente do Departamento de Computação da Universidade Federal de Sergipe (DCOMP/UFS) como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Computação.

São Cristovão, 30 de setembro de 2013.

BANCA EXAMINADORA:

_____

Profº Marco Túlio Chella, Doutor
Orientador
DCOMP/UFS

_____

Profº Luiz Brunelli, Doutor
Universidade Federal de Sergipe

_____

Profº Ricardo José Paiva de Britto Salgueiro, Doutor
Universidade Federal de Sergipe

Resumo

Esse trabalho propõe uma arquitetura de rede aplicada à Domótica. Essa rede implementa os conceitos de Internet das Coisas e Arquitetura Orientada a Serviço. Os elementos da rede, assim como sua comunicação, foram analisados, discutidos e projetados. Foi implementado um caso de uso para demonstrar na prática a arquitetura proposta. Os objetivos do trabalho foram alcançados com sucesso, os nós pertencentes à rede foram projetados e analisados, a comunicação foi discutida e o caso de uso funcionou como o esperado. O custo do trabalho foi considerado baixo visto que foram adquiridos quatro nós, sendo um de fronteira, um mestre e dois locais, além do gasto com os módulos de comunicação sem fio e componentes para os nós locais. O trabalho, além de explorar problemas não solucionados nos conceitos propostos, fornece uma implementação de rede que pode ser reproduzida em outros trabalhos e assim possibilitar que outros projetos possam ser implementados com um custo reduzido.

Palavras-chave: Domótica, Internet das Coisas, *WebService RESTful*, Sistemas Embarcados

Abstract

This work describes the proposal of a network architecture applied to Domotics. This network implements the concepts of Internet of Things and Service Oriented Architecture. The network elements, as well as their communication, was analyzed, discussed and designed. It was implemented a case study to demonstrate in practice the proposed architecture. The objectives were successfully achieved, the nodes of the network are designed and analyzed, the communication was discussed and the case study work as expected. The total cost did was considered inexpensive since there was four nodes acquired, one border, one master and two locals, beyond the cost of the wireless modules and components to the local nodes. The work, besides exploring unsolved problems in the proposed concepts, provides a network deployment that can be replicated in other studies and thus enable other projects to be implemented with a reduced cost

Keywords: Domotics, Internet of Things, *Web*Service RESTful, Embedded Systems

Índice de Figuras

Índice de Tabelas

Sumário

Lista de Siglas

6LoWPAN - IPv6 over Low Power Wireless Personal Area Networks

API - Application Programming Interface

AURESIDE - Associação Brasileira de Automação Residencial

DHCP - Dynamic Host Configuration Protocol

HTTP - Hypertext Transfer Protocol

IoT - Internet of Things

IP - Internet Protocol

IPSO - Internet Protocol for Smart Objects

NFC - Near Field Communication

NPM - Node Package Manager

OSI - Open Systems Interconnection

RFID - Radio-Frequency IDentification

RIP - Routing Information Protocol

SOA - Service Oriented Architeture

SPI - Serial Peripheral Interface

SoC - System on Chip

URI - Universal Resource Identifier

USB - Universal Serial Bus

WSAN - Wireless Sensor and Actor Networks

1. Introdução

Os sistemas de automação residencial estão em pleno crescimento, mas ainda destoam quando comparados a outros segmentos da automação, como por exemplo as linhas de produções da indústria onde se tornou mais comum a presença de supervisores e operadores de máquinas do que a de trabalhadores.

Outro segmento da automação onde é visível um grande avanço é o da automação automobilística. Vidros elétricos são características básicas em um carro popular e carros de luxo apresentam características como direção elétrica, limpadores de para-brisa automáticos, navegador GPS, marchas automáticas e velocidade controlada.

Enquanto nesses segmentos a automação acompanha a evolução da tecnologia, na automação residencial muitas características parecem ser de um futuro distante, a população ainda não crê numa casa onde as luzes ascendem ao passar uma pessoa, janelas que se fecham automaticamente quando está chovendo ou em cortinas que controlam a sua abertura de acordo com a luminosidade.

O custo da automação no setor industrial se transforma em investimento, visto que o que será gasto em automação será recuperado em um número menor de funcionários ou numa maior quantidade da produção. No setor automobilístico o preço da automação é diluido no preço do automóvel se apresentando como adicionais desejados pelos consumidores. No setor imobiliário a visão de automação como investimento para uma redução no uso da energia elétrica, melhor aproveitamento do tempo dos residentes ou como adicionais de conforto para os moradores ainda não se equivale à visãonos outros setores.

Um dos grandes problemas da automação residencial se apresenta na falta de comunicação entre as soluções tradicionais. Esse problema se apresenta de forma clara que a Associação Brasileira de Automação Residencial (AURESIDE) (2013) divulga a importância de se contratar um profissional denominado “Integrador de Sistemas Residenciais” que assume a tarefa de projetar previamente a integração ds soluções tradicionais definindo o cabeamento, infra-estrutura, plataformas de automação e relação de equipamentos necessários.

Este trabalho visa apresentar uma arquitetura de uma rede de nós sensores/atuadores sem fio utilizando conceitos de Internet das Coisas, Domótica e Arquitetura Orientada a Serviços. Essa rede deverá auxiliar na comunicação entre nós e

na comunicação de nós com o meio externo, a Internet, fornecendo tarefas de automação residencial na forma de serviços.

1.1. Objetivos

1.1.1. Objetivos Gerais

O objetivo geral desse trabalho é a demonstração de como a Internet das Coisas e a Arquitetura Orientada a Serviços podem agregar valor à Domótica, tanto no projeto de uma casa automatizada como no desenvolvimento de produtos para a mesma.

1.1.2. Objetivos Específicos

Os objetivos específicos são:

- Projetar nós sensores e atuadores orientados a serviços – Foram projetados nós que fornecem e consumem serviços de outros nós;
- Conectar esses nós em uma rede doméstica sem fio de baixo custo – Os nós projetados serão conectados através de uma tecnologia sem fio de baixo custo para permitir a comunicação entre eles, assim podendo fornecer e consumir serviços;
- Integração da rede doméstica à Internet implementando a Internet das Coisas – A rede local será conectada à Internet através de um nó especial, assim os nós locais poderão se comunicar com nós de outras redes com a mesma arquitetura ou com outros serviços, além de poder fornecer serviços a outros dispositivos conectados à Internet.
- Implementação da rede projetada em um caso de uso – A implementação mostra os conceitos abordados pela arquitetura de rede em funcionamento sendo possível realizar análises de desenvolvimento da rede projetada.
- Fornecer um projeto de baixo custo que possa ser replicado com facilidade para auxiliar futuros trabalhos.

1.2. Metodologia

O desenvolvimento do trabalho se iniciou com a pesquisa dos conceitos abordados no trabalho, primeiramente sobre a automação residencial e suas aplicações e em seguida sobre a Internet das Coisas.

Após a pesquisa foi realizado um estudo de arquiteturas existentes, tanto relacionadas diretamente com Internet das Coisas como relacionada com as redes de computadores, os conceitos essenciais das arquiteturas foram analisados. Também foi estudada a arquitetura orientada a serviço e suas vantagens que poderiam ser aproveitadas no trabalho.

Após a pesquisa e análise de trabalhos relacionados foi realizado o projeto e implementação de uma arquitetura de rede que utilizasse os conceitos de Automação Residencial, Internet das Coisas e Arquitetura Orientada a Serviço.

2. Fundamentação Teórica

2.1. Redes de Computadores

Kurose e Ross (2006) dizem que a Internet provavelmente é o maior sistema de engenharia já criado pela humanidade. A todo momento milhares de computadores e sistemas computacionais estão conectados e trocando informações. Tanenbaum afirma que as principais conquistas tecnológicas do Século XX se deram no campo da aquisição, processamento e distribuição de informações (2003).

2.1.1. *Hardware* de rede

Em um projeto de redes, há questões técnicas que precisam ser estudadas. Primeiramente os principais tipos de tecnologia de transmissão, também os modos de operação da rede além de uma escala de tamanho para poder classificá-la e saber quais problemas irão surgir.

Quanto às tecnologias de transmissão, Tanembaum (2003) afirma que os tipos de tecnologia mais usados nas redes de computadores são os links de difusão e os links ponto a ponto.

Os links de difusão possuem um canal de transmissão compartilhado. As máquinas ligadas a essa rede enviam pacotes com um campo de endereço do destinatário. Todas as máquinas irão receber esse pacote e caso ele esteja endereçado para a mesma, ela irá processá-la. Caso contrário o pacote será ignorado.

Já os links ponto a ponto, as redes são compostas por várias conexões entre pares de máquinas. O pacote de uma máquina poderá passar por algumas máquinas intermediárias antes de chegar ao seu destino.

Quanto aos modos de operação, é possível classificar três principais: *broadcast*, *unicast* e *multicast*. O *broadcast* é quanto uma máquina envia um pacote e ele é processado por todas as máquinas da rede. O *unicast* é quando uma máquina envia um pacote diretamente a outra. Já o *multicast* é quando uma máquina envia um pacote para um grupo de computadores da rede.

Quanto ao tamanho, as redes podem ser classificadas em: redes locais,

metropolitanas ou geograficamente distribuídas.

As redes locais, as LANs, possuem um tamanho reduzido e permite a análise do pior caso de desempenho. São redes que se localizam em um único edifício ou campus universitário. Além do seu tamanho reduzido, as LANs geralmente são conectadas em um cabo que se conecta com todas as máquinas. Essas suas características permitem que se construa uma rede local com topologias que em redes maiores seriam inviáveis.

As redes metropolitanas, as MANs, se originaram das redes de distribuição de TV e sua topologia é fortemente influenciada por elas. Essas redes possuem uma caixa de junção que recebe a informação de um *head end* e envia para as casas.

As redes geograficamente distribuídas, as WANs, abrangem uma grande área, frequentemente um país ou um continente. Ela é formada por um conjunto de máquinas chamadas *hosts*, cuja finalidade é executar os programas dos usuários. Esses *hosts* são interconectados em uma sub-rede que pertence e é operada por um provedor de serviços da Internet. A WAN possui dois elementos distintos, as linhas de transmissão e elementos de comutação. As linhas de transmissão transportam *bits* entre duas máquinas e os elementos de comutação recebem dados de uma linha e devem escolher a linha de saída que precisam encaminhá-los. Os elementos de comutação também podem ser chamados de roteadores, eles conectam uma LAN à WAN.

2.1.2. *Software* de rede

Para facilitar o projeto de uma rede, grande parte delas é organizada como uma pilha de camadas. O objetivo das camadas é abstrair a complexidade da implementação fornecendo uma interface amigável para as camadas superiores.

A camada n de uma máquina se comunica indiretamente com a camada n de outra máquina. As regras e convenções usadas nesse diálogo são chamadas de protocolo da camada n. Como dito anteriormente as camadas de mesmo nível não se comunicam diretamente, elas transferem dados para a camada inferior até chegar à camada física e ao chegar no destino toda a informação é passada para a camada superior até chegar ao mesmo nível da camada de origem.

Uma lista de protocolos usados por um sistema, um proocolo para cada camada, é chamado de pilha de protocolos. O conjunto de camadas e protocolos é chamado de arquitetura de rede.

2.1.3. Modelo OSI

O objetivo do modelo OSI foi a padronização internacional dos protocolos empregados nas diversas camadas (Day e Zimmermann, 1983). A proposta desse modelo foi desenvolvida pela ISO.

O modelo possui sete camadas, cada um com uma função bem definida:

- **Camada física** – Possui como objetivo enviar *bits* de uma máquina para outra, suas questões mais comuns são a tensão usada, tempo de transmissão e a possibilidade ou não de transmissão nos dois sentidos.
- **Camada de enlace de dados** – Sua principal tarefa é transformar um canal de transmissão bruta em uma linha que pareça livre de erros de transmissão para a camada de rede.
- **Camada de rede** – Ela controla a operação da sub-rede. A principal questão do projeto é determinar como é feito o roteamento dos pacotes da origem até o destino. O roteamento pode ser feito através de tabelas estáticas, definidas no início da conversação ou até mesmo por tabelas altamente dinâmicas que se alteram a depender do congestionamento da rede.
- **Camada de transporte** – Ela deve aceitar os dados da camada acima dela, se for necessário separar em unidades menores e assegurar que todos os fragmentos chegarão à outra extermidade.
- **Camada de sessão** – Sua principal função é permitir que os usuários de diferentes máquinas estabeleçam sessões entre eles. Dentre suas tarefas é possível citar o controle de diálogo, manter o controle de quem deve transmitir a cada momento, e a sincronização, verificação periódica de transmissão para permitir que, em caso de falha, as máquinas continuem de onde parou.
- **Camada de apresentação** – Ela se preocupa com a sintaxe e semântica das informações transmitidas. Sua função é gerenciar estruturas de dados abstratas e permitir a definição e intercâmbio de estruturas de dados de nível mais alto.
- **Camada de aplicação** – A camada de aplicaçao contém uma série de protocolos necessários para os usuários. Geralmente esses protocolos possuem uma finalidade específica, como por exemplo: FTP para transferência de arquivos, SMTP para transferência de e-mails e HTTP para acesso a documentos.

2.1.4. Modelo TCP/IP

Diferentemente do modelo OSI, o modelo TCP/IP foi estudado após a existência da rede ARPANET (Que originou a Internet). O modelo é bastante ligado aos protocolos padronizados causando uma certa resistência à mudança de tecnologias.

O modelo TCP/IP possui 4 camadas:

- **Camada física e de enlace** – No modelo TCP/IP a camada física e de enlace formam uma camada apenas e não é muito bem especificada. Essa camada permite o envio de pacotes IPs pela rede.
- **Camada de rede** – Durante o desenvolvimento da rede ARPANET pelo Departamento de Defesa dos EUA, um requisito de projeto foi que a rede funcionasse mesmo que alguns elementos de rede fossem destruídos de uma hora para outra, ou seja, que os pacotes pudessem seguir outra rota a depender da disponibilidade. O objetivo dessa camada é que uma máquina possa enviar pacotes para outra máquina de qualquer ponto da rede para qualquer outro ponto e permitir que eles trafeguem independente pela rede, mesmo que eles cheguem em uma ordem diferente.
- **Camada de transporte** – Assim como no Modelo OSI, a finalidade da camada de transporte é permitir que os hosts de origem e de destino mantenham uma conversação. Os dois protocolos definidos no Modelo TCP/IP são o TCP e o UDP. O primeiro permite que um fluxo de bytes sejam entregues sem erros, ou seja, eles serão entregues na ordem correta. O segundo, o UDP, não fornece esse tipo de garantia.
- **Camada de aplicação** – A camada de aplicação, assim como no Modelo OSI, possui protocolos de nível mais alto, dentre eles o TELNET (protocolo de terminal virtual), o FTP (Protocolo de transferência de arquivos) e o SMTP (protocolo de correio eletrônico).

2.1.5. HTTP

O protocolo HTTP é definido no RFC 1945 e no RFC 2616. É bastante usado por fazer parte da *Web*. É implementado em uma arquietura cliente-servidor. Seu fucncionamento se baseia no acesso a objetos através de uma URL, o cliente faz uma requisiçãode um objeto ao servidor e obtém uma resposta.

O Protocolo HTTP utiliza o TCP como protocolo da camada de transporte pelas garantias que o mesmo fornece, assim ele não precisa se preocupar se um pacote chegou corretamente ao destino.

2.1.5.1. Formato da Mensagem HTTP

As mensagens HTTP são escritas em texto ASCII comum, assim é possível lê-la e entendê-la de forma relativamente fácil. Como o protocolo HTTP se baseia numa arquitetura cliente servidor, há uma requisição e uma resposta.

A requisição HTTP é composta por três categorias de linhas, são elas: linha de requisição, linha de cabeçalho e corpo de entidade.

A linha de requisição é única na requisição e possui as informações de método requisitado, URL do objeto acessado e a versão do protocolo, separados por espaços em branco. Após a linha há uma quebra de linha indicando o começo do cabeçalho.

Após a linha de requisição há as linhas de cabeçalho, cada uma delas possui um nome e um valor separados por um espaço, logo após há uam quebra de linha. Uma linha em branco sinaliza o fim do cabeçalho e o início do corpo de entidade.

O corpo de entidade não possui uma padronização quanto à forma, ela vai depender de aplicação para aplicação.

A resposta HTTP também possui três categorias, sendo duas delas as linhas de cabeçalho e corpo de entidade. A diferença está na primeira linha, que na resposta é chamada de linha de estado.

A linha de estado da resposta HTTP possui as informações de versão do protocolo, o código de estado e uma frase referente ao código do estado.

As informações das Linhas de requisição, de estado e de cabeçalho são separadas por espaços em branco.

2.1.5.2. Principais métodos HTTP

Os principais métodos HTTP são apresentados a seguir:

- GET – O método GET requisita a informação de um objeto no servidor, esperando como resposta o conteúdo daquele objeto.
- POST – O método POST envia informações para serem processadas em um objeto no servidor.

- PUT – O método PUT envia uma nova representação do objeto pra o servidor.
- DELETE – O método DELETE remove o objeto do servidor
- HEAD – O método HEAD é semelhante ao método GET, porém retorna apenas o cabeçalho, sem retornar o corpo de entidade.
- OPTIONS – O método OPTIONS retorna os métodos HTTPs disponíveis no servidor.

2.2. Sistemas Embarcados

Os sistemas embarcados podem ser definidos como sistemas cuja capacidade de processamento de informações está embarcado na unidade de processamento, ou seja, o *software* do sistema está interno à estrutura do mesmo sem a necessidade de uma memória externa. (OLIVEIRA e ANDRADE, 2011)

2.2.1. Memória

O componente que caracteriza um sistema ser embarcado é a presença de uma memória com os dados do programa. Sem a mesma o processador necessitaria de uma memória externa para poder realizar suas funções, descaracterizando o sistema embarcado.

2.2.1.1. Características da Memória

Dentre as características da memória é possível destacar:

- **Tempo de Acesso** – O tempo de acesso é o tempo necessário para a memória ser acessada e realizar uma operação de leitura ou gravação.
- **Capacidade** – A capacidade da memória é a quantidade efetiva de dados que podem ser armazenadas no seu interior.
- **Não Volatilidade** – É a capacidade da memória não perder duas informações quando a energia é cessada.
- **Tempo de Latência** – É o tempo mínimo que se deve esperar entre cada operação de leitura ou escrita na memória.

2.2.1.2. -- Tecnologias de Memórias

Dentre as tecnologias de memórias é possível citar 6 delas:

- **Memória RAM** – É uma memória volátil. Ela permite o acesso a qualquer ponto da memória sem precisar percorrê-la, diferentemente das antigas memórias em fita.
- **Memória ROM** – É uma memória não volátil. Ao deixar de ser energizada não perde as informações, porém ela é apenas de leitura, sendo programada pelo fabricante apenas.
- **Memória PROM** – É um tipo de memória ROM que permite a gravação de dados, mas apenas uma vez. É programada pelo projetista.
- **Memória EPROM** – Também é um tipo de ROM que permite a gravação de dados, porém eles podem ser apagados algumas vezes por processos especiais.
- **Memória EEPROM** – Semelhante à EPROM, porém pode-se apagar os dados através de processos elétricos.
- **Memória FLASH** – É programável e apagável eletricamente, possui velocidade superior, mas também um maior custo.

2.2.1.3. -- Memória de Programa

O *software* do sistema embarcado se encontra na memória de programa. Geralmente a esma é uma memória FLASH por possibilitar diversas gravações e leituras com uma velocidade superior à EEPROM.

2.2.1.4. -- Memória de Dados

A memória de dados armazena os valores temporários do sistema embarcado, sejam ele registros, variáveis ou qualquer outro espaço temporário. Pela não necessidade de se manter os valores ao se desligar o sistema é utilizada uma memória RAM para a memória de dados.

2.2.1.5. -- Memória de Armazenamento

Nos sitemas embarcados ainda há também uma memória de armazenamento

responsável por guardar dados e parâmetros importantes para o sistema embarcado, informações estas que se deseja manter mesmo que o sistema seja desligado. Por essa necessidade a memória de armazenamento é uma memória EEPROM.

2.2.2. - Arquitetura Harvard x Von Neumann

As memórias precisam ser ligadas ao processador por meio de barramentos. Em sistemas computacionais há basicamente dois tipos de arquitetura que definem a quantidade e ligações dos barramentos, são elas: Arquitetura de Von Neumman e a Arquitetura Harvard.

A arquitetura de Von Neumann é caracterizada pela presença de um barramento único para instruções e dados. O processador precisa terminar de receber um dado para poder ler outra instrução.

A arquitetura Harvard é caracterizada pela presença de um barramento para as instruções e um barramento para os dados. Essa arquitetura permite que instruções sejam processadas enquanto os dados são recebidos pelo processador.

2.2.3. - Tecnologias de Projeto de Processadores RISC x CISC

A diferença de processadores RISC e CISC são no número e aprofundamento das instruções.

Os processdadores RISC (Reduced Instruction Set Computing) possuem um número reduzido de instruções, geralmente elas realizam operações básicas necessárias para se programar como operações de soma, subtração, condicional e desvio de instrução.

Já os processadores CISC (Complexo Instruction Set Computing) possuem instruções mais complexas, essas instruções a mais podem ser realizadas por mais de uma instrução de uma arquitetura RISC. Um exemplo é oferecer as operações básicas de soma com as opções de realizar entre um endereço de memória e um registrador, entre dois endereços de memória ou entre dois registradores.

2.2.4. - Microprocessadores e Microcontroladores

Os microprocessadores e microcontroladores realizam o processamento de

informações digitais. A diferença básica entre eles é que o microprocessador necessita de periféricos externos como uma memória que contenha a memória de programa, o barramento que irá ligá-os e qualquer outro periférico que seja necessário à aplicação.

Já o microcontrolador possui suas próprias memórias internas, periféricos essenciais ao seu funcionamento e periféricos opcionais para generalizar o propósito do microcontrolador.

2.2.5. - Alimentação

A alimentação de sistemas embarcados é muito importante pois diferente de um grande sistema, não necessariamente ele estará ligado à rede elétrica. Geralmente um sistema embarcado precisará ser alimentado por pilhas ou baterias. É necessário levar em consideração as problemáticas da alimentação no projeto do sistema embarcado

2.2.6. - Comunicação

A comunicação entre dois dispositivos potencializa o poder de processamento de informações.A necessidade de comunicação pode surgir por um deles tem um melhor desempenho para processar volumes de dados ou por conseguir guardar mais informações ou controlar um sistema a distância. Seja qual for o motivo, a comunicação flexibiliza as aplicações, porém as torna mais complexas.

2.2.6.1. Paralela

A comunicação paralela se caracteriza pela possibilidade de enviar mais de um *bit* de informação simultaneamente. As portas paralelas presentes em um computador pessoal transferem 8 *bits* simultaneamente. Eram bastante comuns em impressoras antigas.

2.2.6.2. Serial

Diferentemente da comuicação paralela, a comunicação serial envia *bits* sequencialmente, o fato de enviar apenas 1 *bit* facilita o interfaceamento por precisar de um barramento menor do que a porta paralela.

2.2.6.2.1 UART

A comunicação UART (Universal asynchronous receiver transmitter) é uma comunicação serial assíncrona bidirecional, ou seja, não há a presença do clock para controlar o tempo de envio e recebimento de informações e é possível enviar e receber uma informação ao mesmo tempo. o barramento para a comunicação UART é composto por um fio RX, chamado de receptor, um fio TX, chamado de transmissor e o fio de referência.

Para que a comunicação seja possível é necessário que os dois dispositivos saibam a qual taxa irão enviar e receber os dados, essa taxa é medida em *bits* por segundo, geralmente a taxa de envio e recebimento é 9600 *bits*/s.

Outra informação necessária para a comunicação é saber quantos *bits* possui a unidade de informação enviada. Esse valor pode variar de 7 à 9, geralmente a unidade de informação são 8 *bits*, completando 1 byte.

Outras informações se devem à paridade, *bit* de parada. O primeiro é um *bit* que informa se o número total de *bits* 1 enviados é par ou ímpar, a depender da configuração. O segundo é o *bit* que informa o início e fim de bloco, pode ser zero ou um.

2.2.6.2.2 SPI

A comunicação SPI (Serial Peripheral Interface) é uma comunicação síncrona do tipo mestre-escravo que se baseia na troca de informações (MICROCHIP).

O tamanho do barramento é de pelo menos 4 fios, um deles transporta a informação do mestre para o escravo (MOSI), outro transporta a informação do escravo para o mestre (MISO), o terceiro é o clock do mestre e o quarto é a referência, o terra. Outros fios podem ser necessários para a escolha de qual escravo o mestre manterá uma comunicação (SS).

A cada ciclo de clock o mestre envia um *bit* de informação ao escravo e recebe dele um *bit* de informação também, o tamanho da unidade de informação pode variar, geralmente sendo 8 *bits*, assim como na comunicação UART.

2.2.6.2.3 I2C

A comunicação I2C também é uma comunicação síncrona, porém não permite o envio e recebimento de informações síncronas. O tamanho do barramento é de 3 fios, sendo um deles para dados, outro para o clock e outro de referência. A comunicação I2C utiliza uma forma de endereçamento o que permite a comunicação entre diversos dispositivos com um barramento menor que o SPI.

2.2.6.3. USB

A comunicação USB surgiu para suprir a necessidade de ligar em um mesmo computador vários periféricos e que ao conectá-los o computador pudesse reconhecê-los sem configurações adicionais.

A comunicação USB possui um barramento com 4 fios, dois de alimentação e 2 de transmissão, D+ e D-. A informação está presente na diferença de tensão entre os dois fios de transmissão.

A tecnologia USB não é de domínio público. Os fabricantes precisam pagar uma licença para eles e seus produtos. Dessa forma eles adquirem um VENDOR ID e PRODUCT IDs relacionados a eles. Para não ser necessário um investimento desse tipo para pequenos projetos é possível utilizar IDs fornecidos por fabricantes ou para classes de produtos específicos, como mouses e teclados, assim não é necessário pagar uma licença para utilizar o USB de forma legal.

Uma alternativa bastante utilizada é a simulação de uma porta serial através de uma porta USB. CIs dedicados fornecem essa possibilidade bastando instalar o driver correspondente.

2.2.6.4. Ethernet

A comunicação Ethernet permite a interconexão de redes locais. Ela é bastante usada para a transferência de pacotes IPs podendo fazer parte da pilha de protocolos TCP/IP. Ao se implementar a comunicação Ethernet é implementado duas camadas do Modelo OSI, são elas: Camada Física e Camada de Enlace.

2.2.6.5. Sem Fio

Outra possibilidade de comunicação é a comunicação sem fio. A troca de informações é feita através de ondas. Várias são as soluções existentes como por exemplo os mais comuns WiFi e Bluetooth.

2.3. Automação Residencial

A automação residencial e predial originou-se dos conceitos utilizados em automação industrial (SENA, 2005), porém as necessidades de cada um desses mercados são distintas. Enquanto na automação industrial é necessário que os equipamentos sejam imunes às falhas, possuam respostas rápidas e possuam alta precisão, na automação residencial essas condições podem ser amenizadas, mas cresce a necessidade de interfaces amigáveis e intuitivas e acabamento superior.

O termo Domótica vem da junção da palavra latina "Domus", casa, com "Robótica", controle automatizado de algo.

Um marco importante para a automação residencial foi o lançamento dos primeiros módulos inteligentes chamados X-10 na década de 70. A característica mais marcante nesses módulos é a utilização da própria rede elétrica como canal de comunicação utilizando um protocolo homônimo. Posteriormente, na década de 80, houve a popularização dos computadores pessoais (PCs) com interfaces amigáveis e operações fáceis apresentando novas possibilidades de automação. O final da década de 90 é responsável por várias novidades para o mercado de automação residencial, dentre elas o telefone celular e a Internet que despertam grande interesse aos consumidores pela sua facilidade.

A automação residencial objetiva o aumento da eficiência e da qualidade de vida, o uso eficaz da energia e de outros recursos naturais. Dentre as possibilidades da automação residencial podem ser citados: Aparelho de ar condicionado, sensores de presença, videoporteiro eletrônico, câmeras de segurança, sensores de portão eletrônico, etc. Essas possibilidades podem ser vistas na Figura 1.

Figura 1: Possibilidades da Automação Residencial (SENA, 2005, p. 14)

Outra aplicação da automação residencial que ganhou destaque é a automação inclusiva. Utilizar a automação residencial para prover sistemas que facilitem um deficiente, ou um idoso a ter uma vida mais autônoma e independente. Dentre sistemas que podem ser fornecidos, pode-se citar sinalizadores sonoros com frases previamente gravadas auxiliando deficientes visuais, sinalizadores visuais e luminosos que diferenciam toques de campainha e de telefone auxiliando deficientes auditivos, central de monitoramento pessoal que fornece à pessoa um botão que pode ser acionado caso seja necessário uma ajuda auxiliando tanto deficientes quanto idosos.

Uma grande novidade da automação residencial é a do controle de sistemas da casa através de um tablet conectado à Internet, onde o aparelho provê uma interface amigável e intuitiva em um aparelho portável e de outras funcionalidades.

2.4. Domótica

A domótica está para a automação residencial assim como a robótica está para

a automação industrial. A robótica trata de sistemas mecânicos motorizados controlados de forma automática. Há vários níveis de controle sendo um deles, por exemplo, em nível de tarefa onde é passado ao robô uma tarefa e o robô a realizará. A domótica trata de sistemas mecânicos com o objetivo de automatizar as tarefas do lar.

Os sistemas de domótica atualmente oferecem as soluções apresentadas na introdução desse trabalho. As soluções mais simples envolveriam a resposta de um atuador a um evento disparado pelo usuário, como por exemplo o ligar de uma lâmpada através de um botão de um controle remoto do usuário, mas essas soluções podem se tornar mais elaboradas.

Grande parte dos sistemas de domótica trabalham com o conceito de *triggers* (disparos), ou seja, eventos que acontecem e terão um resultado, semelhante à interação com uma GUI (Interface gráfica de usuário). As soluções oferecem o disparo de eventos de forma manual, aleatória e programada.

Essa variedade fornece várias possibilidades de configuração automática da casa e da utilização da mesma, fornecendo uma solução para a interação do próprio usuário com a casa, no caso de disparos manuais; interação da casa com o ambiente, no caso de disparos programados a partir de condições, como por exemplo o ligar das lâmpadas da área externa ao chegar a noite e de uma simulação de atividades na casa quando a família está em férias, no caso dos disparos aleatórios.

Além dessa habilidade de tratar com disparos, as soluções geralmente oferecem o acesso remoto às funcionalidades da residência devido à facilidade de se monitorar e controlar a casa através de um dispositivo móvel. Outra funcionalidade seria a integração com o sistema de segurança para a gravação de ambientes quando necessário e respostas da casa aos intrusos, como ligar todas as lâmpadas da casa, travar portas etc.

Uma funcionalidade que vem crescendo é a do reconhecimento de voz. Com a tecnologia atual é possível utilizar comandos de voz de forma natural e controlar a casa, como por exemplo recolher as persianas ao dizer: “Abrir janela”.

A compatibilidade possui certo problema devido à necessidade das soluções de poderem suportar diversas tecnologias como por exemplo suportar a tecnologia X10, pioneira na automação residencial, ou a recente Z-Wave, além de outros protocolos. Quanto ao sistema operacional várias soluções suportam versões do Windows, do iOS e o Android, visto que são os mais populares dentre a maioria dos potenciais clientes (TOPTENREVIEWS, 2013).

Aqui no Brasil há empresas desenvolvendo tecnologia para automação

residencial, como por exemplo a Iluflex (ILUFLEX), as soluções propostas também envolvem os pontos citados anteriormente como os disparos, suporte a vários sistemas operacionais, reconhecimento de voz e outras possíveis integrações.

2.5. Internet das Coisas

O conceito de Internet das Coisas ainda não é bem definido, Atzori et al (2010) definem três orientações para as várias visões possíveis da Internet das Coisas. Um dos motivos para se haver várias interpretações é a falta de uma padronização quanto a esse conceito e a união de duas palavras de semânticas diferentes, a palavra "Internet" leva à visão orientada por Redes da IoT, a palavra "Coisas" traz a tona "objetos" genéricos, quaisquer, trazendo consigo outra visão para a mesma palavra. Além da diferença entre essas duas palavras, o problema com as inúmeras possibilidades de coisas conectadas gera uma nova visão orientada pela semântica.

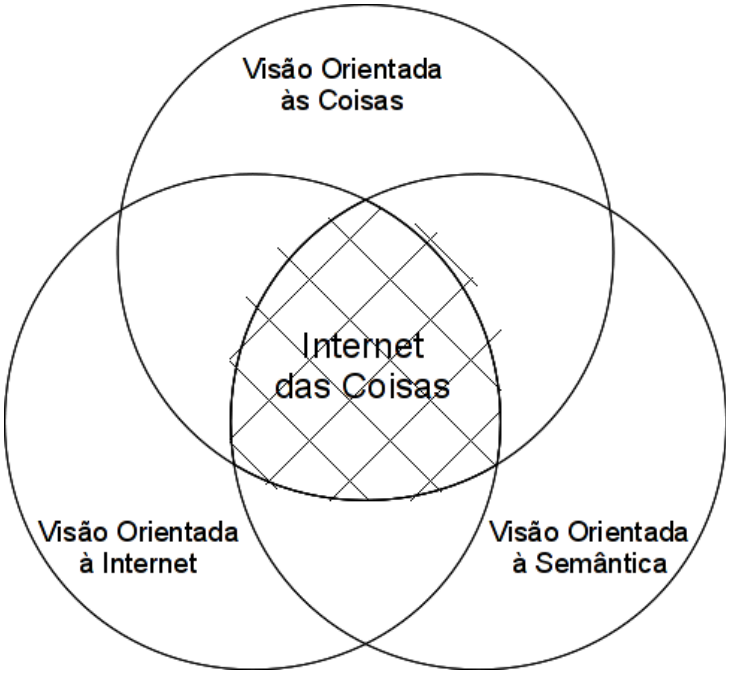

Figura 2: As três visões da IoT, adaptado de Atzori et al (2010)

A primeira definição vem da perspectiva "das Coisas", considera coisas como etiquetas identificadores em Rádio-Frequência (Radio-Frequency IDentification – RFID). O termo "Internet das Coisas" seria uma rede de etiquetas identificando produtos que poderiam ser monitorados a qualquer momento (AUTO-ID LABS).

Em uma visão mais ampla a IoT não pode se limitar apenas em etiquetas identificadoras, eles fazem parte de algo maior. Além do RFID, Comunicações de campos próximos (*Near Field Communications* – NFC) e Rede de sensores e atuadores sem fio (*Wireless Sensor and Actor Networks* – WSAN) seriam componentes atômicos que ligariam o mundo real ao mundo digital (PRESSER; GLUHAK 2009).

Sob a perspectiva "da Internet", há a definição de IP para Objetos Inteligentes (IP for Smart Objects – IPSO). Essa definição é originária de uma aliança homônima (IPSO) (DUNKELS; VASSEUR, 2008) fundada por 25 empresas para promover o protocolo IP como tecnologia de conexão para objetos inteligentes. Um dos frutos dessa aliança é o 6LoWPAN (HUI; CULLER; CHAKRABARTI, 2009), uma adaptação do protocolo IP incorporando o IEEE 802.15.4. O IEEE 802.15.4 se refere as redes pessoais sem fio com baixíssimo consumo de energia e uma redução considerável em velocidade (IEEE, 2013), o 6LoWPAN adapta o IP para sistemas embarcados permitindo que um protocolo utilizado em vários outros dispositivos possa ser usado também para objetos inteligentes. *Internet 0* segue uma visão similar, reduzindo a complexidade do IP para atingir o "IP sobre qualquer coisa" (GERSHENFELD; KRIKORIAN; COHEN, 2004).

A ideia por trás da perspectiva "da Semântica" está na grande quantidade de coisas conectadas na Internet do futuro, sendo assim é necessário resolver futuros problemas gerados pela IoT como a representação, armazenamento, busca e organização de informações. Isso gera soluções de modelos de informações apropriados para as coisas (TOMA; SIMPERL; HENCH, 2009).

A tecnologia usada para identificação, sensoriamento e comunicação na grande maioria das pesquisas é o Rádio. A redução do tamanho, peso, consumo de energia e custo permitem que seja adicionado o Rádio a qualquer objeto permitindo a comunicação entre esses objetos. Seu uso pode ser tanto em etiquetamento (RFID tags) como em comunicação, a exemplo do Bluetooth, Wi-Fi, 6LoWPAN e Zigbee.

Outra tecnologia bastante utilizada pela IoT é a utilização de *middlewares*, eles abstraem as diferenças entre *hardwares* para uma aplicação sendo uma importante camada para permitir o agrupamento de coisas com diferentes *hardwares*. A arquitetura mais usada para os *middlewares* é a Arquitetura Orientada a Serviço.

As aplicações da Internet das Coisas são bastante variadas, mas é possível agrupá-las em 5 tipos: Transporte e Logística, Saúde, Pessoal e Social, Ambientes Inteligentes e Futurístico. As aplicações da Internet das Coisas passam pelo

monitoramento de mercadorias, monitoramento de objetos roubados, táxis robôs, coleta de informações biológicas, casas automáticas além de várias outras aplicações (ATZORI et al, 2010).

Algumas questões em aberto sobre a IoT referem-se às padronizações, dentre elas: suporte móvel, nomeação dos objetos, protocolos de transporte, autenticação, integridade de dados, privacidade.

2.6. Arquitetura Orientada a Serviços

Arquitetura orientada a serviço (*Service oriented Architeture* - SOA) é um estilo arquitetural que suporta o paradigma de orientação a serviços (FUGITA e HIRAMA, 2012, p. 10). Por sua vez, a orientação a serviços é um paradigma de construção e integração de soluções de *software* compostas por elementos modulares chamados serviços. (Erl, 2007, apud FUGITA e HIRAMA, 2012, p. 7).

Segundo Fugita e Hirama (2012, p. 8) o serviço, no contexto de SOA, é uma unidade de *software* que tem como propósito desempenhar uma função específica e que pode ser fornecido por um provedor e utilizado por um consumidor.

O serviço pode ser comparado com uma classe, da orientação a objetos. Fazendo essa comparação podemos dizer que um serviço fornece operações a um consumidor assim como as classes possuem métodos associados a ela que podem ser invocados.

SOA é uma arquitetura onde são modelados provedores de serviços que serão consumidos por clientes, esses serviços podem ser consumidos de forma remota. Dentre os benefícios de SOA está o encapsulamento do serviço, ou seja, basta saber como é feita a requisição desse serviço e como ele irá fornecer o resultado do mesmo, a sua implementação e tecnologia são insignificantes perante o consumidor.

3. Revisão Bibliográfica

3.1. Domótica

Dentre os trabalhos relacionados a este na área de domótica é possível citar o trabalho de Miori et al (2006). Nesse trabalho eles desenvolvem uma padronização para o fornecimento de funcionalidades da automação da casa na forma de serviços. É discutida a arquitetura proposta por eles além da proposta do DomoML. O DomoML é uma linguagem de marcação que descreve os serviços de um dispositivo no formato XML.

Aiello e Dustdar (2008) apresentam uma discussão sobre a evolução da domótica atrelada a arquitetura orientada a serviço. Eles propõe uma arquitetura para uma casa onde a saúde de um idoso é monitorada e implementam essa arquitetura em dispositivos heterogêneos.

3.2. Internet das Coisas

Gomez-Goiri et al (2011) apresenta um estudo sobre a utilização de TS (Triple space) como camada de aplicação para a Internet das Coisas. Esse trabalho mostra também a preocupação em como consumir os serviços fornecidos pelos dispositivos conectados à Internet.

Buckl et al (2009) mostra em seu trabalho uma forma de conectar a Internet às redes embarcadas. Utilizando uma abordagem separatista seu trabalho apresenta também como unir esses dois mundos ao traduzir uma requisição de serviço em uma mensagem interpretável pelos dispositivos conectados à rede local.

Shelby et al (2003), em seu trabalho, apresenta a proposta do nanoIP. O nanoIP é uma minimalização da camada de rede da pilha TCP/IP. Além do nanoIP também é proposto o nanoUDP e o nanoTCP que são a minimalização dos protocolos UDP e TCP, respectivamente. Seu trabalho apresenta a comparação quantitativa do cabeçalho dos protocolos quando comparados com os protocolos tradicionais.

Castellani et al (2010) descrevem uma implementação de Internet das Coisas na Universidade de Padova, Itália. Na implementação foi utilizado o protocolo BWS

(Binary *Web* Service) que é uma forma binária, reduzida, do REST, mas compatível com o protocolo HTTP. Também foi utilizado o protocolo 6LoWPAN para permitir que a rede fosse acessada através do IPv6.

4. Desenvolvimento

A arquitetura do sistema consiste basicamente de uma rede de sistemas sem fio. Cada sistema da rede compõe um nó, podendo ser classificado como nó local, nó de fronteira ou nó mestre.

Os nós locais podem fornecer suas funcionalidades como serviços, pode consumir serviços locais ou realizar a tarefa de um roteador.

O nó de fronteira faz o papel de gateway da rede. Ligado à Internet e à rede local ele realiza a tradução do protocolo da rede externa para a rede interna e a tradução contrária também. Essa funcionalidade permite que clientes externos consumam serviços da rede.

O nó mestre está conectado ao nó de fronteira e com os nós locais permitindo a comunicação entre eles.

O formato da rede permite a comunicação entre nós locais sem hierarquia, ou seja, um nó local pode se comunicar com qualquer outro nó da rede. O nó mestre pode se comunicar com qualquer nó da rede e se comunica diretamente com o nó de fronteira. O nó de fronteira, por sua vez, se comunica com o nó mestre e com uma rede externa.

O formato da rede pode ser considerado em malha para a rede local e ponto a ponto entre o nó mestre, o nó de fronteira e a Internet. Uma configuração possível da rede pode ser vista na Figura 8.

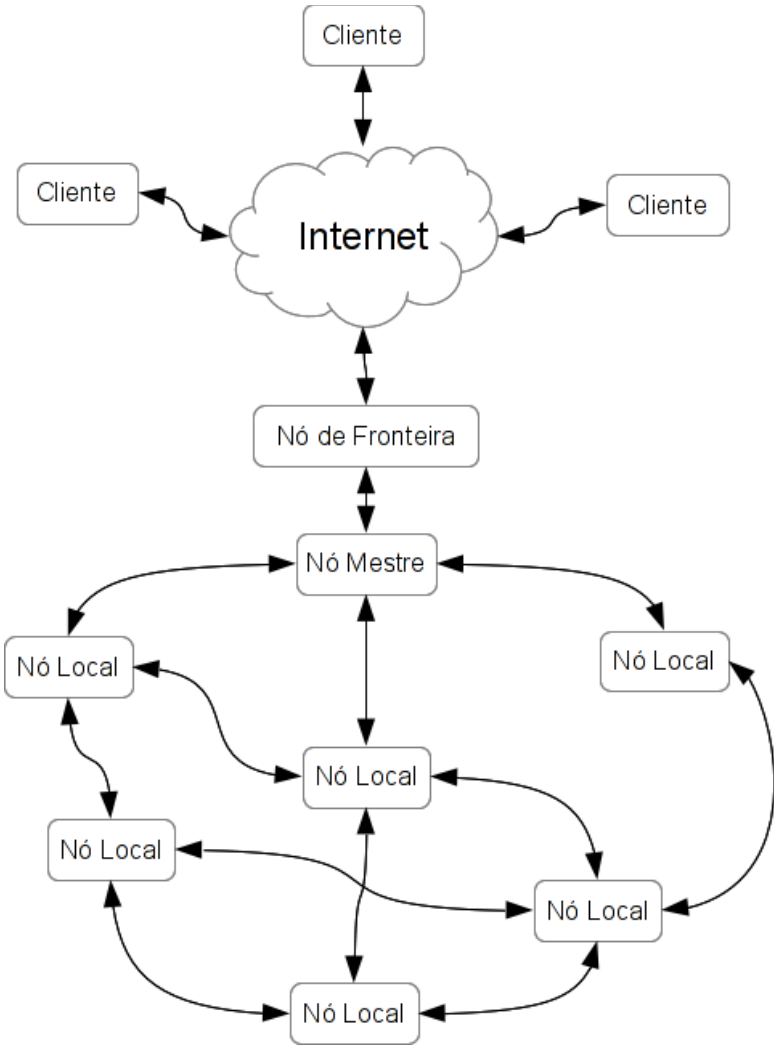

Figura 3: Possível configuração da rede

4.1. Nó local

O nó local da arquitetura proposta é o elemento responsável pelo fornecimento de serviços da rede. Ele se comunica diretamente com outros nós da rede através de um módulo de comunicação sem fio.

Seus componentes são: um microcontrolador, um módulo de comunicação sem fio, sensores e atuadores. Os sensores e atuadores são opcionais, a presença ou não eles servirá como classificação do nó local. Os únicos componentes obrigatórios são: O módulo de comunicação sem fio e o microcontrolador.

Os sensores fornecem informações ao nó, essas informações podem ser usadas internamente ou fornecidas através de serviços. Os atuadores também podem ser usados

internamente ou através de serviços.

O microcontrolador é responsável pela aquisição das informações dos sensores e pelo controle dos atuadores, além disso ele é responsável pelo processamento das requisições e possivelmente de alguma solicitação de serviços. O microcontrolador é responsável por prover a funcionalidade do sistema embarcado que representa o nó local.

O módulo de comunicação sem fio é responsável pela comunicação do nó local com outros nós da rede, ele é melhor discutido nas sessões a seguir.

A classificação dos nós locais pode ser feita a partir de determinadas características, sendo as cinco consideradas nesse trabalho listadas a seguir:

- Nó Sensor X Não-sensor – É caracterizado pela presença, ou não, de sensores;
- Nó Atuador X Não-atuador – É caracterizado pela presença, ou não, de atuadores;
- Nó Cliente X Não-cliente – É caracterizado pelo consumo, ou não, de serviços;
- Nó Servidor X Não-servidor – É caracterizado pelo fornecimento, ou não, de serviços;
- Nó Roteador X Não-roteador – É caracterizado pela retransmissão, ou não, de pacotes.

É importante notar que as características são independentes entre si. Um nó local ser classificado como nó sensor não implica que ele será um nó servidor. A Tabela 1 exemplifica a classificação de possíveis nós locais.

Tabela 1: Classificação de alguns exemplos de possibilidades de nós locais.

	Nó Sensor?	**Nó Atuador?**	**Nó Cliente?**	**Nó Servidor?**	**Nó Roteador?**
Dimmer para lâmpada	Não	Sim	Não	Sim	Sim ou Não
Interface com Usuário	Sim	Não	Sim	Não	Sim ou Não
Sensor de temperatura, umidade e luminosidade	Sim	Não	Não	Sim	Sim ou Não
Roteador	Não	Não	Não	Não	Sim

4.1.1. Tecnologia Escolhida

Para a escolha do *hardware* do nó local foi analisado apenas o microcontrolador. Os sensores podem variar bastante de nó para nó assim como os atuadores e o módulo de comunicação sem fio é discutido em outra sessão por influenciar outras características da rede.

Os três fabricantes de microcontroladores analisados foram a Atmel, Microchip e Texas Instruments. Apenas essas três fabricantes foram analisadas pela facilidade em se conseguir o material de desenvolvimento.

A Atmel possui a linha de microcontroladores AVR, disponíveis em quatro famílias: tinyAVR, megaAVR, XMEGA e Atmel At94k. A plataforma de desenvolvimento mais acessível que usa um microcontrolador dessa fabricante é a plataforma Arduino.

A plataforma Arduino é composta por uma placa de desenvolvimento e uma IDE, ambos de código aberto. Devido a essa natureza livre, há uma comunidade bastante ativa que fornece suporte para diversas aplicações. Uma foto da placa Arduino DUE pode ser vista na Figura 4.

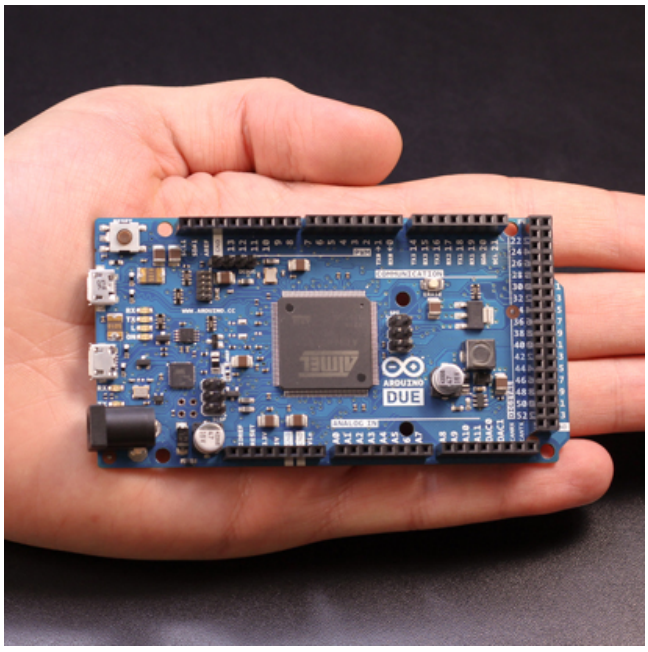

Figura 4: Foto do arduino DUE (ARDUINO).

A Microchip possui uma linha consolidade no mercado de microcontroladores, a linha PIC. Essa linha possui diversas famílias de microcontroladores, como por

exemplo a 12F, 16F, 18F, 30F e 32MX. É possível achar diversos cursos e bibliotecas disponíveis, mas geralmente tem um custo, inclusive para as ferramentas de desenvolvimento.

A Texas Instruments possui o microcontrolador MSP430 como o principal para aplicações menos poderosas. Inspirada no Arduino ela também desenvolveu uma plataforma chamda LauchPad. O *software* de desenvolvimento é originário do mesmo que originou o *software* do Arduino sendo de fácil aprendizado e utilização. Uma foto da placa LaunchPad MSP430 pode ser vista na Figura 5.

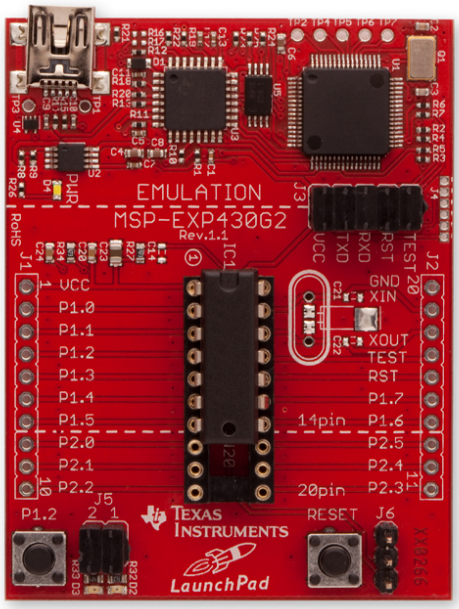

Figura 5: Foto da LaunchPad do MSP430

Todas as três fabricantes possuem microcontroladores com poder de processamento suficiente para o projeto e a custos semelhantes. A tecnologia escolhida foi a do Arduino por possuir um vasto material gratuito na Internet.

4.2. Nó de Fronteira

O nó de fronteira é o responsável pelo interfaceamento da rede local com a rede externa. Sua presença é necessária pois os nós locais não conseguem se comunicar diretamente com a Internet.

O nó de fronteira possui uma conexão com a rede externa e com um nó da rede local, o nó mestre. No no de fronteira as requisições se serviço são recebidas, mapeadas para o protocolo local e enviadas para a rede local para que possa alcançar o nó de destino. O diagrama do nó de fronteira e suas conexões pode ser visto na Figura 6.

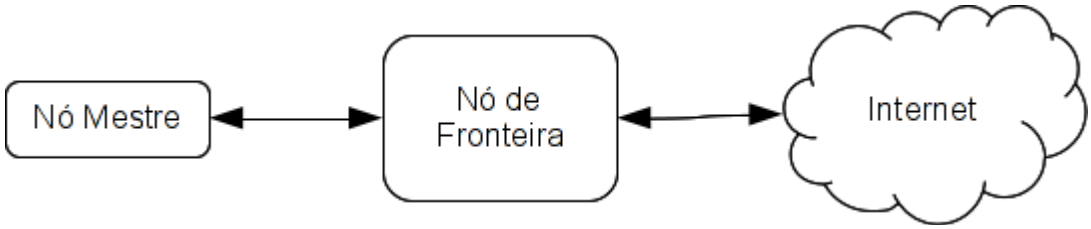

Figura 6: Diagrama do Nó de Fronteira com suas conexões

4.2.1. Tecnologia Escolhida

Para o nó de fronteira é possível utilizar um Computador Pessoal (Personal Computer – PC), um notebook ou até mesmo um servidor, essas seriam as escolhas mais tradicionais, porém são caras, ocupam um espaço relativamente grande e demandam bastante energia.

Outra opção se origina com os processadores ARM. Os processadores ARM são processadores de propósito genérico que podem ter o seu projeto comprado por uma fabricante e utilizado para compor um sistema em um chip (System on Chip – SoC). A popularização desses processadores permite sistemas completos em um espaço reduzido, como um microcontrolador, porém com um desempenho alto, comparável a computadores pessoais menos potentes. Um exemplo claro são os smartphones cuja grande maioria dos processadores utilizam essa tecnologia e possuem clocks na faixa de 1GHz.

Além dos smartphones, esses SoCs permitiram a fabricação de computadores embarcados com periféricos oriundos de PCs, como portas USB, saída de vídeo e audio HDMI, entrada para cartões de memória, etc. Nas versões mais potentes é possível instalar sistemas operacionais antes compilados apenas para processadores x86 e x64 para essas versões do ARM Cortex-A.

Os processadores ARM se dividem basicamente em três categorias: Cortex-M, Cortex-R e Cortex-A; sendo a primeira para aplicações microcontroladas, a segunda para aplicações em tempo real e a terceira para aplicações mais potentes.

Dentre essas placas que utilizam esses SoCs para o processamento podemos citar:

- BeagleBoard, que utiliza um SoC da Texas Instruments
- i.MX53, que utiliza um SoC da Freesclae
- RaspberryPi, que utiliza um SoC da Broadcom
- Arduino Due, que utiliza um SoC da Atmel
- Android 4.0 Mini PC, que utiliza um SoC da Allwinner

Todas as placas utilizam SoC com núcleos ARM, sendo o do Arduino Due o ARM Cortex-M3 e os dos outros o ARM Cortex-A8, por esse motivo o Arduino Due foi descartado dentre as opções por possuir um processador inferior.

Dentre as outras opções a BeagleBoard, o i.MX53 e a RaspberryPi se caracterizam como plataforma de desenvolvimento, por possuir, além dos conectores HDMI, USB e Ethernet, pinos de propósito geral, sendo possível assim o interfaceamento com circuitos eletrônicos de forma mais direta. O Android 4.0 Mini PC possui apenas interface HDMI e USB sendo necessário a comunicação Serial através da USB para a comunicação com um circuito externo.

Quanto ao preço, a BeagleBoard custa U$125,00; o i.MX53 custa U$149,00 (incluindo um display gráfico) e a RaspberryPi modelo B custa U$35,00. O modelo B da RasbperryPi foi analisado devido à presença da interface Ethernet. Analisando os preços, escolheu-se a placa RaspberryPi por custar menos da metade do valor das placas BeagleBoard e i.MX53 respectivamente.

A placa RaspberryPi possui o SoC BCM2835 da Broadcom que segundo a especificação da fabricante possui:

- Processador de aplicação ARM1176JZ-F (Núcleo ARM Cortex-A8)
- Co-processador Multimedia dual core VideoCore IV
- Codificador e Decodificador de video H.264 Full HD 1080p30
- GPU VideoCore de baixa potência e alta performance com OpenGL-ES 1.1/2.0

Com essas informações é possível perceber que o seu SoC possui um processamento razoável, mas permitindo também um processamento de vídeo sendo bastante útil para uma possível interação com usuário, sendo possível inclusive conectar a placa a uma TV. Um bom dispositivo para seu preço de U$35,00.

Além disso o Modelo B fornece duas portas USB para conectar algum dispositivo de interface humano-computador, adaptador Wi-Fi ou Bluetooth, também

possui um conector Ethernet para conectá-lo à Internet através de um cabo. Outro ponto convidativo é o da presença dos pinos de propósito geral que permitem a implementação de uma comunicação serial como UART, SPI ou I2C, comuns em dispositivos eletrônicos.

A possibilidade de se comunicar através de protocolos característicos de circuitos eletrônicos e com a Internet, o bom poder de processamento e a possibilidade de se programar interfaces gráficas para uma interação com o usuário tornam a escolha da placa RaspberryPi interessante.

4.3. Nó Mestre

O nó mestre, como dito anteriormente, é o nó da rede local que se comunica diretamente com o nó de fronteira. Ele pode ser classificado como um nó local especial, que não possui sensores nem atuadores e não se comporta como cliente nem servidor, apenas se comporta como um roteador além da comunicação especial com o nó de fronteira. O diagrama de blocos do nó mestre pode ser visto na Figura 7.

Ele não foi modelado internamente ao nó de fronteira por possuir certa independência, uma vez que ele pode repassar um pacote entre a rede local sem precisar repassá-la para o nó de fronteira.

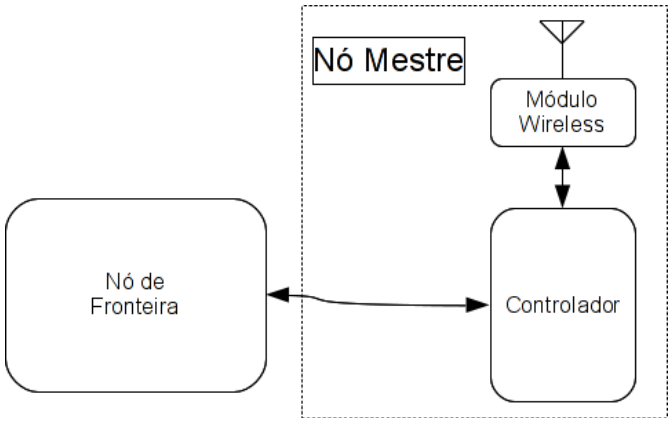

Figura 7: Diagrama de blocos do nó mestre com a conexão com o nó de fronteira

4.4. Módulo de comunicação sem fio

O módulo de comunicação sem fio é o elemento que pertence aos nós locais e ao nó mestre. Fazendo uma analogia com o Modelo TCP/IP ele é responsável pela camada física e de enlace, disponibilizando para a camada superior, a de rede, a possibilidade de enviar pacotes.

4.4.1. Tecnologia Escolhida

As possibilidades de comunicação sem fio são variadas, mas como observado no estado da arte da domótica as escolhas permeiam a rádio frequência. Dentro da rádio frequência há duas frequências de maior abrangência: 433 MHz e 2,4 GHz. A frequência de 433 MHz é utilizada para comunicações curtas e ocasionais, como por exemplo a dos portões controlados por controle remoto ou alarmes de carro. A frequência de 2,4 GHz é mais presente na literatura e abrange o Bluetooth, Wi-Fi, Zigbee e o 6LoWPAN.

O Zigbee e o 6LoWPAN são opções relativamente caras o que desencorajou a escolha deles para a comunicação local. Os dispositivos Wi-Fi de baixo custo possuem uma interface USB e necessitam de um drive, o que não é viável utilizar no microcontrolador escolhido. Outra opção é o módulo de comunicação sem fio da Nordic Semiconductor, o NRF24L01. Seu baixíssimo custo e interface SPI simplificam a utilização do mesmo.

O módulo de comunicação sem fio NRF24L01 possui um custo menor que as outras opções citadas. Além disso sua comunicação com a camada superior é feita através de uma comunicação SPI. Outro ponto positivo é a existência de uma biblioteca para Arduino para utilizar esse módulo. Em poucas horas é possível montar o circuito e experimentar uma comunicação simples entre os dois módulos.

O NRF24L01 utiliza em sua camada física a comunicação via rádio com frequência de 2,4 GHz. Na camada de enlace ele utiliza uma tecnologia denominada Shockburst. Essa tecnologia permite um endereçamento de recebimento de até 5 bytes, controle no envio e recebimento de pacotes e envio de pacotes com até 32 bytes de informação.

Há uma versão melhorada desse módulo, o módulo NRF24L01+, que possui todas as funcionalidades do tradicional, porém com opções de configurações e

redundância a mais.

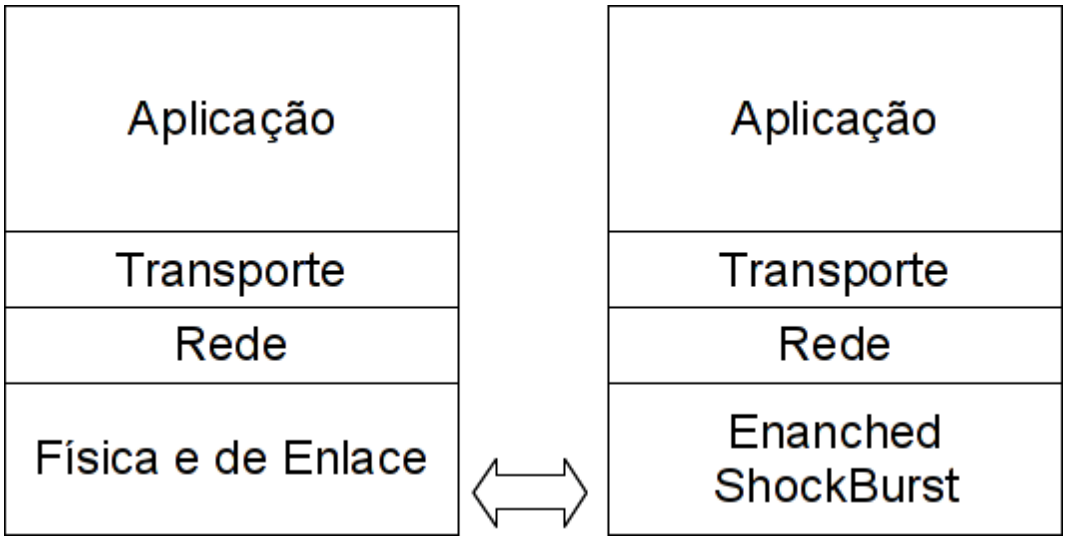

Figura 8: Comparação entre o modelo TCP/IP e a camada representada pelo Módulo NRF24L01

Como é possível ver na Figura 8, o móduo NRF24L01 equivale à camada física e de enlace do modelo TCP/IP, sendo necessário implementar a camada de rede, transporte e aplicação.

4.5. Protocolos da Rede

Como visto anteriormente, o módulo de comunicação sem fio escolhido ocupa a camada física e de enlace, deixando ainda vazia a camada de rede, transporte e aplicação.

Para a camada de rede é preciso fazer o endereçamento. O módulo de comunicação sem fio utilizado permite o endereçamento dinâmico, mas o seu transporte é feito apenas ponto a ponto. Para realizar o roteamento de pacotes é preciso armazenar informações em tabelas, clientes externos também precisam ser endereçados, logo a solução tomada foi de reendereçar os módulos e clientes externos.

Um protocolo é essencial para a comunicação entre dois dispositivos, sem ele é impossível a tranferência de dados coerentes. No modelo OSI há várias camadas de protocolos, o módulo de comunicação sem fio usado nesse projeto implementa até a camada de transporte, com o Enhanced ShockBurst, porém para permitir que os

módulos se comuniquem a um nível de serviço é necessário acrescentar mais uma camada na comunicação entre os mesmos. As sessões seguintes apresentam o projeto e implementação do protocolo usado na camada de aplicação.

4.5.1. Pacote

Como visto anteriormente, o módulo NRF24L01 implementa as camadas: Física, de Enlace, de Rede e de Transporte; Porém na camada de transporte o seu funcionamento é parcial, visto que a mensagem transportada possui apenas o endereço de destino e não o de origem. Essa limitação influencia na falta de informação ao nó destino quanto ao nó de origem para que se possa devolver a resposta.

Como a intenção é permitir a disponibilização de serviços pelos nós da rede, foi necessário a escolha do modelo de serviço a ser implementado, as duas opções mais famosas são os serviços orientados aos recursos, REST, e os serviços orientados às atividades, SOAP.

A escolha da orientação dos serviços foi feita após a análise dos dois modelos consolidados na literatura. Essa análise foi feita no artigo de James Snell, *Web* services orientados a recursos vs. Orientados a atividades (SNELL, James, 2004). A arquitetura REST se apresentou melhor adaptável à realidade da rede. Mesmo que a arquitetura SOAP fosse usada, os serviços estariam divididos entre os nós da rede, com cada nó fornecendo um determinado serviço, logo é mais interessante orientar os serviços a partir dos recursos da rede, ou seja, seus nós.

Os serviços orientado a recursos geralmente utilizam o protocolo HTTP para comunicação, pois os recursos podem ser mapeados através de um Identificador Universal de Recursos (Universal Resource Identifier, URI) e suas operações fazem parte da especificação do protocolo, são elas: GET, POST, PUT e DELETE.

Apesar do HTTP facilitar bastante a implementação de um serviço REST, não é possível utilizá-lo diretamente por restrições dos nós projetados nesse trabalho. O módulo de comunicação utilizado pode transferir pacotes de até 32 bytes, uma mensagem HTTP simples, como a que pode ser vista a seguir, necessitou de 55 bytes. Além disso é possível ver que há muita informação redundante que pode ser compactada, como por exemplo a versão do HTTP, que está ocupando 8 bytes, mas pode ser reduzido para, por exemplo, 2 *bits* e assim representar além das 3 versões que já existiram uma possível versão futura, mesmo que representasse em 4 *bits* e assim

obter uma resolução de 16 versões diferentes seria ocupado apenas 6,25% do espaço gasto com a mensagem HTTP. A requisição usada para essas medições é apresentado a seguir:

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
```

Esse problema do tamanho da mensagem HTTP se deve a característica do protocolo poder ser lido facilmente por humanos, característica que pode ser perdida para permitir uma compactação. Devido a esse problema o protocolo HTTP não se tornou uma possibilidade, sendo a melhor opção na concepção do trabalho o projeto de um protocolo alternativo que se adequasse às limitações presentes das tecnologias escolhidas e pudesse representar as características básicas de uma arquitetura REST.

As subsessões a seguir mostram como as informações de uma requisição e resposta HTTP foram analisadas e quais decisões foram tomadas para a definição do protocolo e como o mesmo foi implementado para a sua utilização nas linguagens de programação C e JavaScript, utilizadas nos nós locais e no nó de fronteira, respectivamente.

4.5.1.1. Requisição

Na Requisição, as informações necessárias em um serviço REST baseado no HTTP são:

- Recurso acessado – Através da URI;
- Método realizado – Entre GET, POST, PUT e DELETE;
- Parâmetros do método – Passados pela URI ou pelo corpo da mensagem.

Neste trabalho apenas essas informações foram levadas em conta para a simplificação do protocolo. O programa que recebe as requisições HTTP é o responsável por obter todas essas informações.

4.5.1.2. Resposta

Na resposta, as informações são levemente diferentes do que na requisição, inclusive na mensagem HTTP. A respota não possui a informação do recurso acessado, pressupõe que o cliente sabe qual recurso ele acessou, nem do método realizado, também pressupõe que o cliente tem consciência do método que ele realizou. Apesar de

não ter essas informações da requisição, a resposta possui outras informações importantes, como:

- Código de estado – Informação da natureza da resposta, se foi um sucesso, se foi falha na requisição, se foi falha no servidor ou até mesmo redirecionamento;
- Conteúdo da Resposta – Informação presente no corpo da mensagem, caso tenha sido uma requisição que espera uma resposta.

Além dessas informações presentes na resposta HTTP, também é importante saber o endereço do cliente para poder respondê-lo, porém essa informação pertence ao protocolo IP. Nesse trabalho a API usada para a comunicação via HTTP se encarrega em saber qual o endereço do cliente para respondê-lo.

4.5.1.3. Projeto do Protocolo

Após a análise das informações necessárias em cada tipo de mensagem HTTP, foi decidido que as seguintes informações iriam compor o protocolo:

- Endereço do Recurso – Identificado pela URI da requisição HTTP;
- Endereço do Cliente – Identificado pelo endereço do Cliente;
- Tipo de Mensagem – Identifica se a mensagem é uma Requisição ou uma Resposta;
- Informação da Mensagem – Caso seja uma requisição, armazena o método, caso contrário armazena a informação do código de resposta;

Antes de se decidir quantos *bits* cada informação poderá ocupar é importante analisar o que cada informação irá armazenar e assim poder escolher um valor justo para o espaço ocupado por cada um. Valor justo esse que possa armazenar os valores necessários para a informação e não ocupe espaço que poderia ser melhor aproveitado.

Levando-se em conta que um mesmo cliente pode requisitar serviços de um mesmo recurso e um intervalo de tempo curto, foi acrescentado a informação de Identificador do Pacote, assim é possível endereçar um pacote unicamente para uma determinada quantidade de pacotes.

Para decidir quantos *bits* cada informação deverá armazenar foi analisado quantos *bits* são realmente necessários e se é possível aumentar o número de *bits*. Além disso foi um requisito que o cabeçalho ocupasse um número inteiro de bytes para facilitar o empacotamento do corpo da mensagem.

Primeiramente os endereços do recurso e do cliente, como endereço de destino

e de partida. O número mínimo de *bits* é difícil de decidir pois um número reduzido de endereços limitaria o número de nós na rede assim como de clientes externos, mas também não é necessário um número muito grande de endereços para esta aplicação, visto que o número de nós de uma casa, com a tecnologia atual, não ultrapassaria algumas centenas de dispositivos, também por se tratar de uma interação limitada do meio externo com o interno, o número de clientes externos simultaneamente não seria muito alto. Para facilitar o armazenamento de informação foi alocado para cada endereço 8 *bits* de informação podendo haver 256 nós locais e clientes externos somados.

Quanto ao tipo de mensagem, só haverá dois tipos, a requisição e a resposta, logo um *bit* basta para representar o tipo de mensagem.

Quanto à informação da mensagem, uma requisição não precisa da informação do código de resposta, assim como a resposta não precisa saber o método que foi chamado, nem as mensagens HTTP guardam essas informações, logo é possível armazená-las em um único espaço no pacote. Como há quatro tipos de métodos usados, basta dois *bits* de informação, porém há cinco classes de respostas, são elas: 1xx – Informação, 2xx – Sucesso, 3xx – Redirecionamento, 4xx – Falha no Cliente e 5xx – Falha no Servidor; Para representar as cinco classes são necessários pelo menos três *bits* de informação.

Para o identificador, como o tipo de mensagem usa um *bit* e a informação da mensagem usa três *bits*, foram reservados quatro *bits*, completando assim mais um byte.

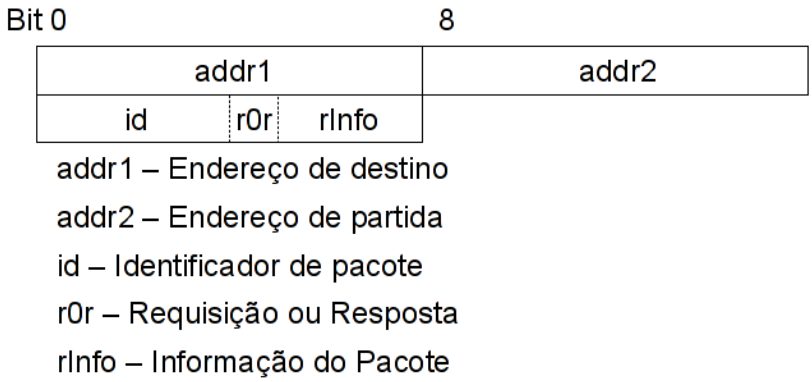

Figura 9: Informações do cabeçalho do pacote projetado neste trabalho

Como é possível ver na Figura Erro: Origem da referência não encontrada, o cabeçalho do pacote ocupa três bytes do total de 32 permitidos pelo módulo NRF24L01, deixando que o corpo possa ocupar até 29 bytes de informação.

4.5.1.4. Codificação e Decodificação

Adquirir as informações de um pacote na forma compacta utiliza várias operações binárias, tornando confusa algumas operações que poderiam ser simplificadas. Para auxiliar nessa tarefa foram desenvolvidas estruturas, tanto na linguagem C como na linguagem JavaScript, que armazenam as informações do pacote, assim elas podem ser acessadas como atributos facilitando a legibilidade e manutenção do código diminuindo o acoplamento, pois mesmo alterando o tamanho das informações é possível manter a mesma estrutura de códigos já desenvolvidos.

Porém é necessário fazer a codificação da estrutura para o pacote e a decodificação do pacote para a estrutura. Para isso foi utilizado operações de deslocamento de *bits*, máscaras e somas, como pode ser visto na Figura Erro: Origem da referência não encontrada e na Figura Erro: Origem da referência não encontrada.

Na Figura Erro: Origem da referência não encontrada é possível ver como se retira informações de um byte. No campo rOr, que está no *bit* número 5, primeiramente faz três deslocamentos de *bit* à direita para que ele fique na sétima posição e logo depois é aplicado a máscara de um *bit*, 0b00000001, assim isola-se o *bit* referente ao campo rOr. A mesma lógica é aplicada aos outros campos.

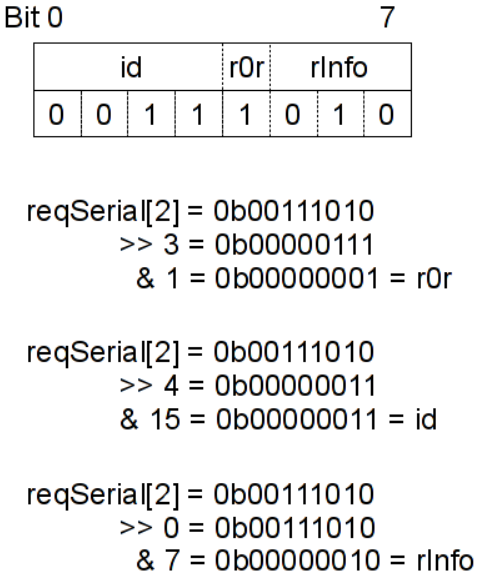

Figura 10: Exemplo de como é possível retirar informações do pacote com deslocamentos e aplicação de máscara

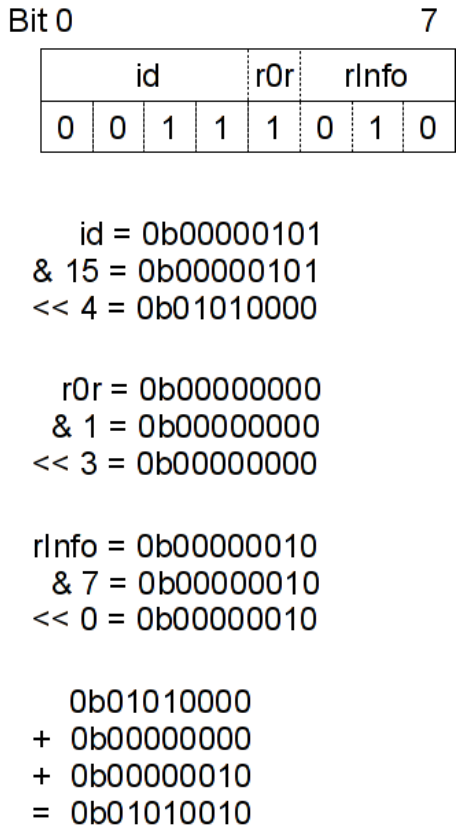

Figura 11: Exemplo de como é possível montar as informações de um pacote com deslocamentos, aplicação de máscara e somas

Na Figura Erro: Origem da referência não encontrada é possível ver a operação inversa apresentada na Figura Erro: Origem da referência não encontrada. Primeiramente aplica-se o dslocamento à esquerda até que a informação fique na sua posição correta e depois somam-se todos os valores dos campos para que se forme novamente o byte original.

Paralelo a essa tarefa foi desenvolvido um programa que dada uma descrição de pacote, fosse gerada uma sequência de instruções que fizessem a codificação e decodificação para a linguagem C.

A Figura Erro: Origem da referência não encontrada mostra um exemplo de descrição do pacote. Cada campo é representado pelo seu nome, seguido de um ponto e vírgula e o número de *bits* que ele ocupa, depois uma quebra de linha. A entrada formatada dessa forma é lida pelo programa e processada para gerar a saída apresentada na Figura Erro: Origem da referência não encontrada. A saída apresenta os comandos necessários para guardar as informações de uma estrutura em um vetor de bytes e os comandos para realizar a operação inversa.

```
addr1:8
addr2:8
id:4
rOr:1
rInfo:3
```

Figura 12: Exemplo de descrição de pacote

```
reqSerial[0] = ((unsigned char)(req->addr1 << 0) & 0xff);
reqSerial[1] = ((unsigned char)(req->addr2 << 0) & 0xff);
reqSerial[2] = ((unsigned char)(req->id << 4) & 0xf0) +
    »    »       ((unsigned char)(req->rOr << 3) & 0x8) +
    »    »       ((unsigned char)(req->rInfo << 0) & 0x7);
reqSerial[3] = ;
req->addr1 = (((unsigned short int)reqSerial[0] >> 0) & 0xff);
req->addr2 = (((unsigned short int)reqSerial[1] >> 0) & 0xff);
req->id = (((unsigned short int)reqSerial[2] >> 4) & 0xf);
req->rOr = (((unsigned short int)reqSerial[2] >> 3) & 0x1);
req->rInfo = (((unsigned short int)reqSerial[2] >> 0) & 0x7)
```

Figura 13: Exemplo de saída do programa que apresenta os códigos para a transcodificação

Já possuindo as instruções para a transcodificação em Linguagem C, é possível adaptá-las para JavaScript, mantendo a mesma lógica, mas com as peculiaridades da linguagem.

4.6. Custo do caso de uso

Tabela 2: Tabela de custos do trabalho

Componente	Preço por unidade	Quantidade	Preço
RaspberryPi	R$ 170,00	1	R$ 170,00
Arduino	R$ 70,00	3	R$ 210,00
Módulo NRF24L01	R$ 20,00	3	R$ 60,00
Cabo USB tipo A/B	R$ 5,00	3	R$ 15,00
LED RGB	R$ 1,10	1	R$ 1,10
LDR	R$ 1,30	1	R$ 1,30
Resistor 10k Ohm	R$ 0,05	1	R$ 0,05
Resistor 1k Ohm	R$ 0,05	3	R$ 0,15
Cabo Ethernet	R$ 8,00	1	R$ 8,00
		Total	R$ 465,60

Ao observar a Tabela 2, é válido notar que do total, 40% é gasto da Raspberry e sua conexão com o roteador da casa. O restante é o gasto para três nós, o que para cada novo nó adicionado da rede o gasto seria de aproximadamente 20% sem contar com os sensores e atuadores do mesmo. É importante destacar que no momento da pesquisa de preço o dólar estava custando 2,23 reais.

4.7. Comunicação

A comunicação entre os elementos da rede não é trivial. Além de ser necessário haver a comunicação entre dois dispositivos, também é necessário que se faça a comunicação ponto a ponto. É possível ver um exemplo de uma requisição externa a um nó local e a quantidade de mensagens transportadas de elemento para elemento, na Figura 14 é possível observar que haverá um atraso decorrente da troca de mensagens entre os elementos da rede e que quanto maior o caminho, maior o atraso. É importante notar que apenas no nó destino a espera do pacote é bloqueante, nos outros elementos o encaminhamento do pacote não fará com que o elemento espere o pacote retornar para continuar seu funcionamento normal. A representação no nó de fronteira informa que a informação da requisição deve ser mantida até que seja respondida, mas o processamento não é bloqueante.

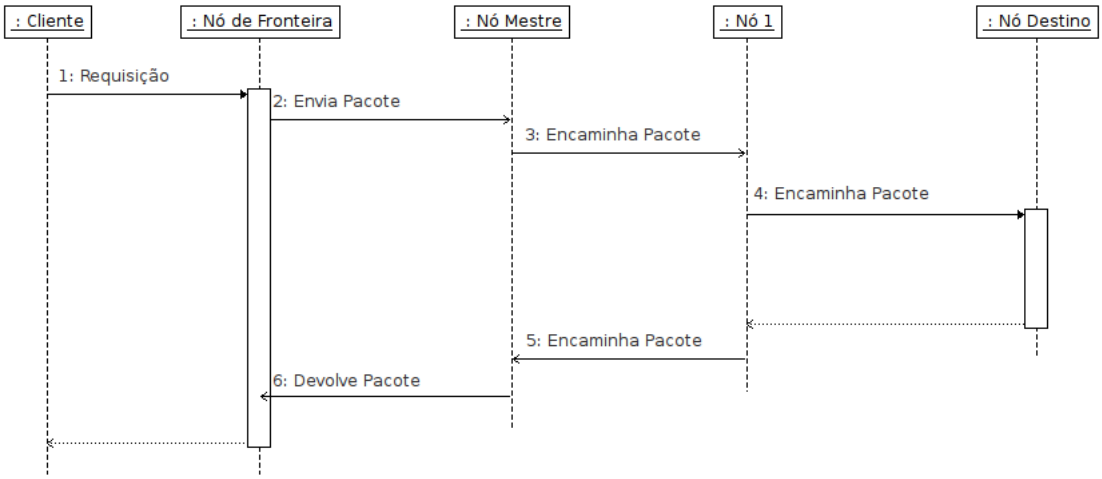

Figura 14: Diagrama de Sequência ilustrando a comunicação entre o cliene e o nó destino

Para o desenvolvimento do projeto foi analisado os passos da comunicação de um cliente externo com um nó local, como seria a entrada, como seria a saída, o tipo de comunicação dentre outras informações de cada passo e após a análise foi feita uma

modelagem e implementação incremental.

Primeiramente foi implementado a comunicação entre o nó de fronteira e o nó mestre, após finalizar essa comunicação foi feita a comunicação entre o nó mestre e um nó local, sequencialmente a comunicação entre um cliente externo e o nó de fronteira e por fim a comunicação entre dois nos locais. Com todas essas comunicações funcionando foi possível uní-las para fechar todos os tipos de ciclo de comunicação da rede.

4.7.1. Nó de Fronteira com Nó Mestre

A primeira parte da comunicação elaborada foi a comunicação entre o nó de fronteira e o nó mestre, no caso do caso de uso, entre a RaspberryPi e o Arduino. Para essa comunicação foi decidido utilizar a comunicação Serial através da pora USB.

A escolha dessa parte da comunicação foi baseada na facilidade proporcionada pela comunicação através da USB. O Arduino já possui uma porta serial conectada à porta USB da placa, além de uma biblioteca completa para comunicação. A RaspberryPi permite a instalação de bibliotecas decomunicação serial. Ao conectar o Arduino na RaspberryPi, por causa do driver do Arduino, ele já é reconhecido como uma pota Serial.

O teste realizado nesse nível de comunicação foi o envio de um caractere pelo Nó de Fronteira e o recebimento de um número inteiro representando a porta analógica do Nó Mestre.

O código utilizado no Arduino implementou o diagrama de atividades representado na Figura 15. Como é possível perceber o programa é simples porém consegue verificar a funcionalidade no envio e recebimento de dados pela porta serial.

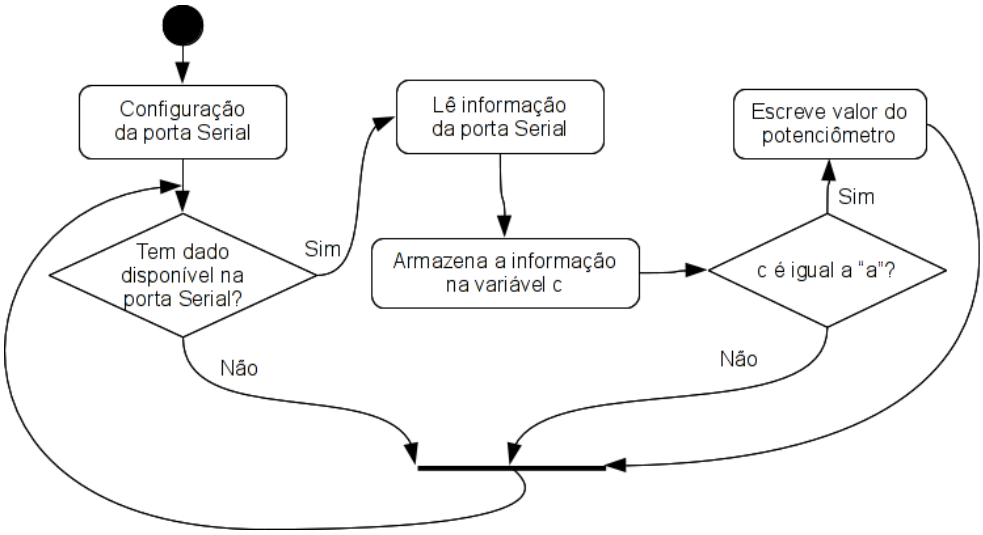

Figura 15: Diagrama de atividades representando o código que o Arduino executa no teste da comunicação entre o nó de fronteira e o nó mestre

O código utilizado na RaspberryPi foi implementado utilizando o NodeJS e implementou o diagrama de atividades representado na Figura 16. O NodeJS é um interpretador JavaScript que pode ser executado fora do navegador e ser utilizado como servidor. Ele possui, na instalação padrão, pacotes para implementar um servidor HTTP, para a comunicação serial foi instalado o pacote serialport disponibilizado no Node Package Manager, NPM.

Basicamente o nó de fronteira envia a cada segundo um caractere e ao receber a informação do nó mestre, apresenta na tela. Apesar de simples, esse código consegue mostrar se a porta serial está funcionando corretamente e em conjunto com o código utilizado no Arduino pode-se ver o funcionamento da comunicação entre os dois.

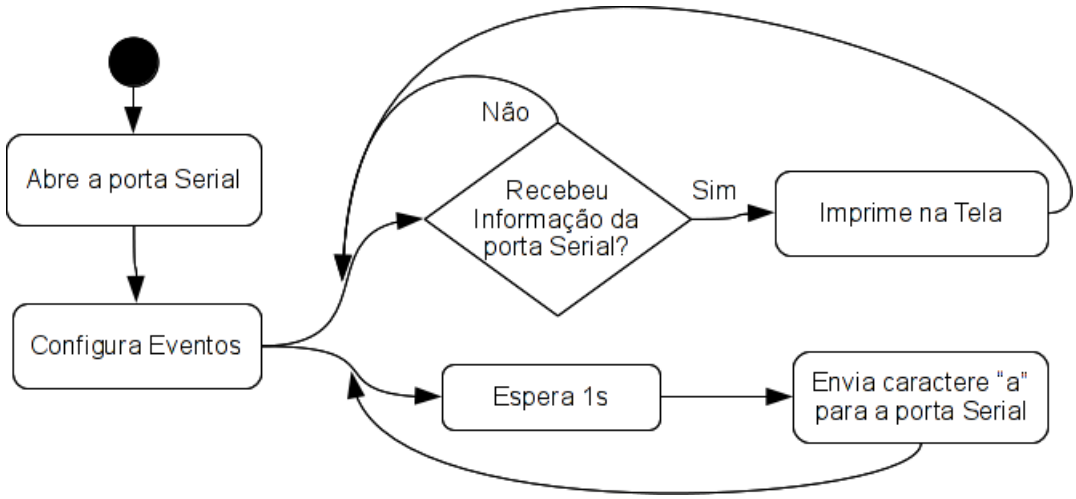

Figura 16: Diagrama de atividades representando o código executado pela RaspberryPi para o tesde de comunicação entre o nó de fronteira e o nó mestre

4.7.2. Nó Mestre com Nó local

A comunicação entre o nó mestre e o nó local se faz por meio do módulo de comunicação sem fio, que nesse trabalho está sendo usado o NRF24L01. Para o teste de comunicação foi usado um código fornecido junto com a biblioteca usada no Arduino para esse módulo. O código citado implementa uma forma de PING, ou seja, há um servidor que espera algum pacote e ao receber, devolve o pacote imediatamente, assim o cliente pode guardar o tempo de envio e de chegada e medir o tempo gasto para o pacote ir e voltar.

O código no servidor funciona da seguinte forma: primeiramente ele se configura adequadamente, abre a sua porta serial, configura o módulo dentre outras configurações; logo após espera o recebimento de um pacote, ao receber o pacote altera o endereço de envio e retransmite o pacote para o cliente, então espera por um novo pacote.

O código no cliente também se configura adequadamente em um primeiro instante, mas antes de enviar o pacote ele guarda o tempo atual, em milisegundos, e configura o endereço de envio para o servidor, depois de enviar o mesmo espera o pacote do servidor e apresenta na porta serial o tempo gasto entre a ida e a volta do PING.

O teste é importante para verificar se os módulos estão conseguindo se comunicar antes de executar uma comunicação mais complexa. Os códigos utilizados nesse teste podem ser encontrados na pasta de exemplos da biblioteca MIRF disponibilizada para o Arduino no link https://github.com/aaronds/arduino-nrf24l01/tree/master/Mirf.

4.7.3. Cliente remoto com Nó de Fronteira

A comunicação entre um cliente remoto e o nó de fronteira é feito através de um servidor http. O nó de fronteira roda um servidor http que irá receber requisições de serviço modeladas em requisições HTTP e adaptá-las à rede local, ao receber uma resposta responder ao cliente como uma resposta HTTP.

Para implementar a comunicação entre o cliente remoto e o nó de fronteira foi implementado um servidor http simples em NodeJS que extrai algumas informações da requisição e responde em forma de texto ao cliente. Nesse caso o cliente é simulado

pelo programa curl. O programa curl permite a requisição de métodos HTTP através da linha de comando, assim é possível realizar as requisições escolhendo os parâmetros adequados para os testes.

Os campos adquiridos da requisição foram: Método, IP Remoto, URL, Corpo da Mensagem e Cabeçalho. Esses campos são os principais para se implementar um *web*Service RESTFUL sendo suficientes para se processar uma requisição e respondê-la.

O comando executado para realizar a requisição foi:

```
curl -X POST -d \
"{\"usuario\":\"Fernando\",\"senha\":\"123deoliveira4\"}" \
localhost:8080 -H "content-type:application/json"
```

E o resultado pode ser visto a seguir:

```
127.0.0.1
{'user-agent': 'curl/7.26.0',
  host: 'localhost:8080',
  accept: '*/*',
  'content-type': 'application/json',
  'content-length': '49'}
/
POST
Fernando
123deoliveira4
{"usuario":"Fernando", "senha":"123deoliveira4"}
```

As informações, de cima para baixo, são: IP remoto, cabeçalho da requisição, URL requisitada, informação “usuario” do corpo da mensagem, informação “senha” do corpo da mensagem e o corpo da mensagem enviado.

4.7.4. Nó local com Nó local

Para a comunicação entre nós locais foram utilizados três arduinos, um representando o nó mestre e os outros nós representando nós locais. Diferentemente dos códigos usados na comunicação entre o Nó Mestre e o Nó Local, foram necessárias modificações no código de exemplo fornecido pela biblioteca. As modificações foram necessárias para se alterar o endereçamento e implementar um roteamento.

Primeiramente foram alterados os endereços dos nós, o endereçamento foi realizado respeitando o diagrama apresentado na Figura 17. No mesmo diagrama é possível perceber como foi realizada a comunicação, do nó mestre para o nó ponte, para o nó servidor e voltando.

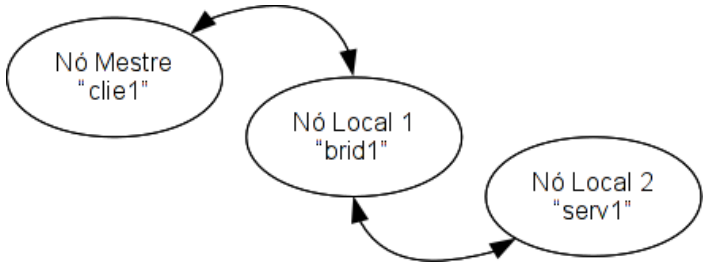

Figura 17: Diagrama de Blocos representando a comunicação no teste entre os Nós locais

Nesse teste o nó "clie1" envia um pacote com a informação do tempo em milisegundos para o nó "brid1" que por sua vez verifica se o primeiro byte é igual a 1, caso negativo envia o pacote para o nó "serv1". Este atribui ao primeiro byte o valor 1 e envia o pacote novamente para o nó "brid1" que agora, ao verificar o valor do primeiro byte, envia-o para o nó "clie1".

Em medições do ping é possível observar o aumento da latência do envio, decorrente do repasse dos pacotes, na comunicação direta a latência observada possui uma média aproximada de 10 microssegundos, já com a comunicação nessa implementação a latência observada alcançava valores na faixa de 1 milissegundo.

4.7.5. Endereçamento e Roteamento

O endereçamento e roteamento nesse trabalho não tiveram muito enfoque, sendo a implementação dos mesmos de forma estática, porém utilizando as informações necessárias do pacote projetado neste trabalho.

O endereçamento dos nós locais é feito de forma estática indo dos valores 1 até o número de nós locais, sendo o valor 0 exclusivo para o nó mestre e os valores acima do número de nós da rede alocados para clientes externos.

O roteamento é feito de forma estática, não possuindo ainda uma tabela de roteamento configurável. De forma simplificada quando um nó recebe um endereço entre 1 e o número de nós locais, ele tenta repassar ao nó mais próximo que pode entregar esse pacote e quando for outro valor de endereço ele redireciona o pacote para o nó mestre, onde o mesmo pode ser o nó destino ou pode responder para clientes externos.

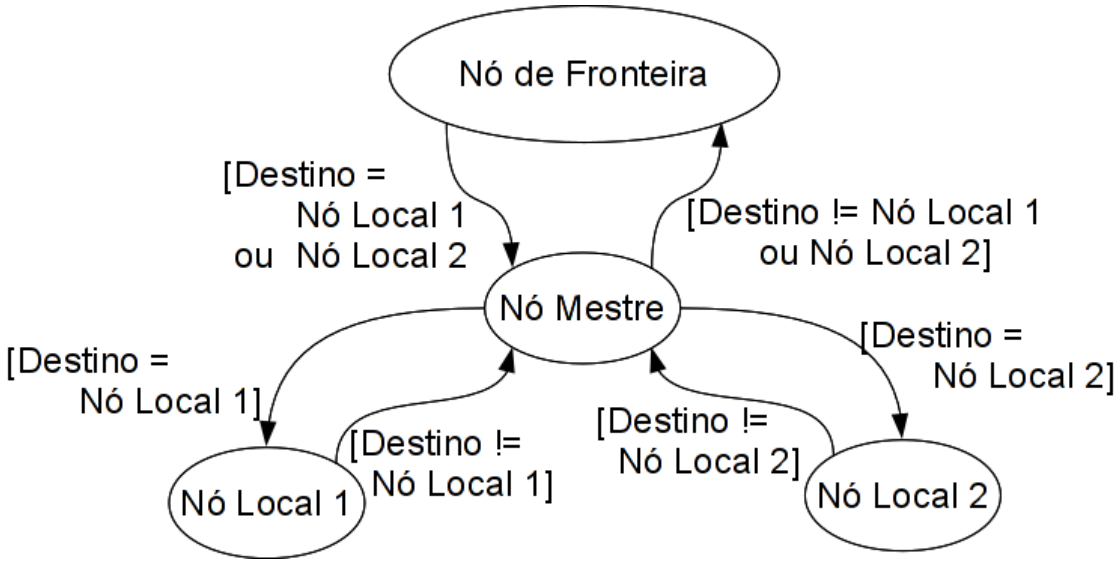

Figura 18: Diagrama exemplificando o funcionamento do roteamento em uma rede hipotética

5. Caso de uso

O caso de uso desenvolvido no trabalho apresenta um cenário simples que ilustra uma automação residencial. O sistema controla as cores de uma lâmpada colorida, seja pela interação com o usuário ou automaticamente pela luminosidade do ambiente.

Primeiramente foi pensado em como estruturar os nós. Como já mencionado no parágrafo anterior, um nó local, chamado node1, ficaria responsável por controlar uma lâmpada colorida, nesse trabalho representado por um LED RGB, como também é um requisito para o projeto, esse nó local forneceria um serviço para alterar a cor da lâmpada. Esse nó não precisaria consumir nenhum serviço externo.

Outro nó local, chamado node2, seria responsável pelo controle automático da luminosidade, ou seja, ao detectar um brilho reduzido o mesmo poderia consumir o serviço do nó local que controla a lâmpada e assim alterar a luminosidade da mesma.

Além dos dois nós locais também foi necessário implementar o nó mestre e o nó de fronteira.

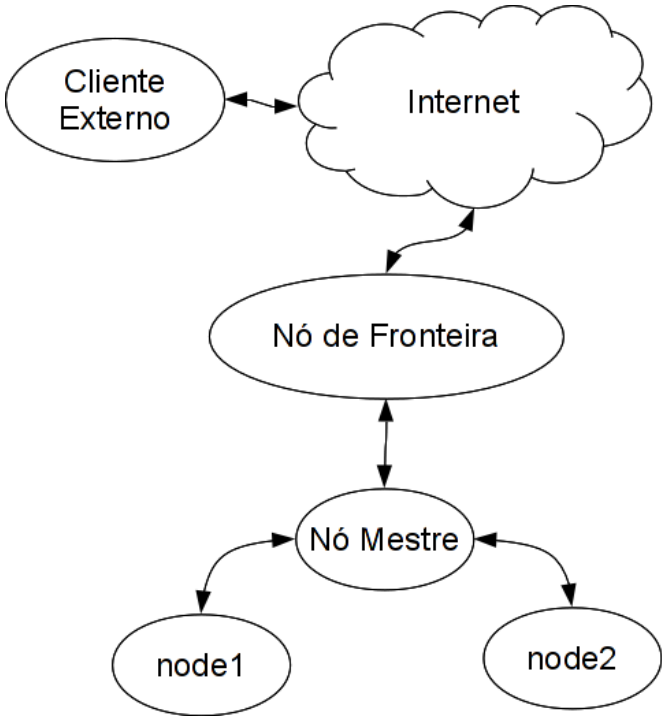

Figura 19: Diagrama da rede implementada no caso de uso

A Figura 19 apresenta o diagrama da rede implementada no caso de uso. A rede local possui dois nós, um nó mestre, o nó de fronteira e um cliente externo. A Tabela 3

apresenta as classificações dos nós da rede local do caso de uso.

Tabela 3: Classificação de alguns exemplos de possibilidades de nós locais.

	Nó Sensor?	**Nó Atuador?**	**Nó Cliente?**	**Nó Servidor?**	**Nó Roteador?**
Nó Mestre	Não	Não	Não	Não	Sim
node1	Não	Sim	Não	Sim	Não
node2	Sim	Não	Sim	Sim	Não

5.1. Serviços fornecidos pela rede

Os serviços disponibilizados pela rede se concentram nos dois nós locais. O primeiro, node1, disponibiliza o serviço para controle de cor da lâmpada. O segundo, node2, disponibiliza o serviço para controle automático da cor da lâmpada.

Node1 fornece: Controle de cor do LED RGB, Parâmetros: Cor Vermelha, Cor Verde, Cor Azul, Retorno: Cor atual do LED.

Node2 fornece: Ativação de controle de cor automática, Parâmetros: Ligado ou Desligado, Retorno: Retorna 1 se estiver em modo automático e 0 caso contrário.

É possível ver, na Figura 20 o arduino ligado ao módulo de comunicação sem fio e ao LED RGB.

A Tabela 4 apresenta os serviços fornecidos pelos nós da rede, cada só fornece dois recursos, cada um com dois métodos.

Tabela 4: Serviços fornecidos pela rede

Nó local	**Recurso do Nó**	**Método**	**Entrada**	**Saída**
node1	corLED	GET	---	Cor em RGB
		POST	Cor em RGB	Cor em RGB
node2	controleAuto	GET	---	1 ou 0
		POST	1 ou 0	1 ou 0

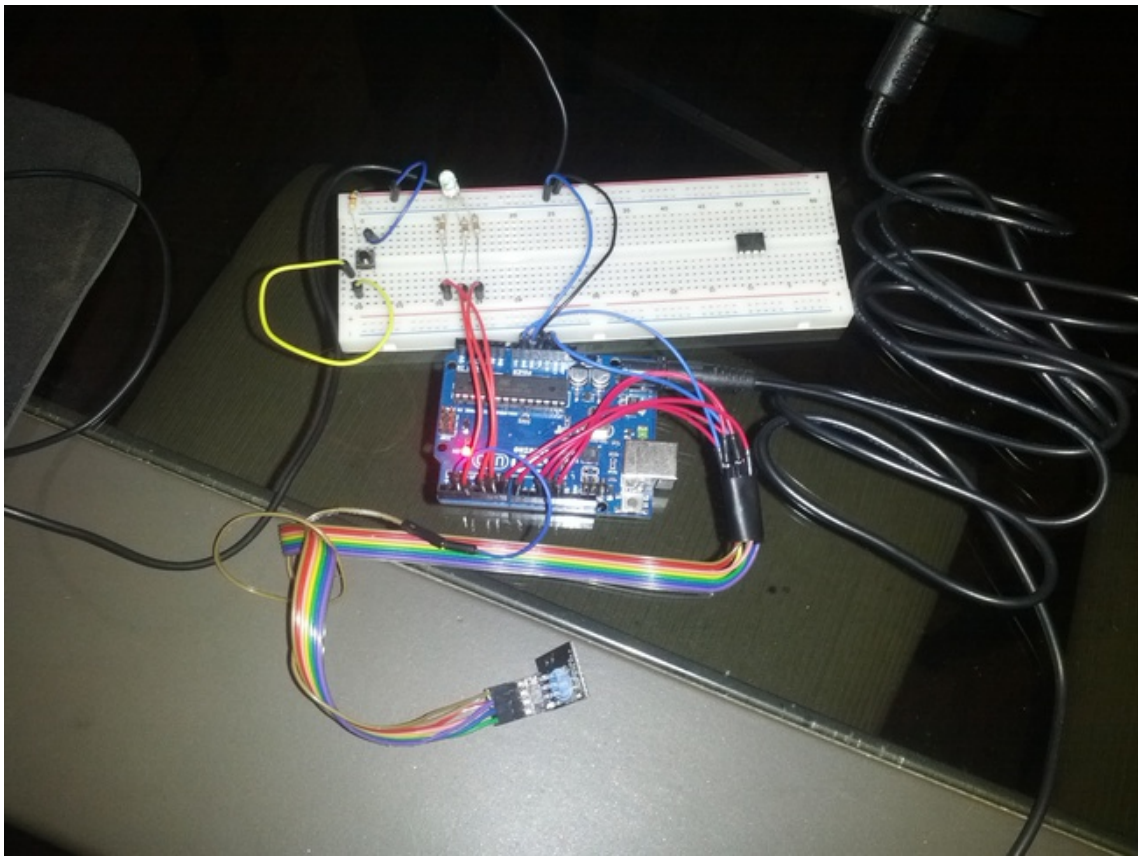

Figura 20: Foto do node1.

A Figura 21 apresenta o diagrama esquemático do nó node1. Nele o LED RGB está representado por três LEDs distintos com os ânodos conectados. É importante notar que os LEDs estão ligados à saídas denominadas PWM que permitem controlar a energia enviada. Com saídas PWM é possível controlar a luminosidade de LEDs.

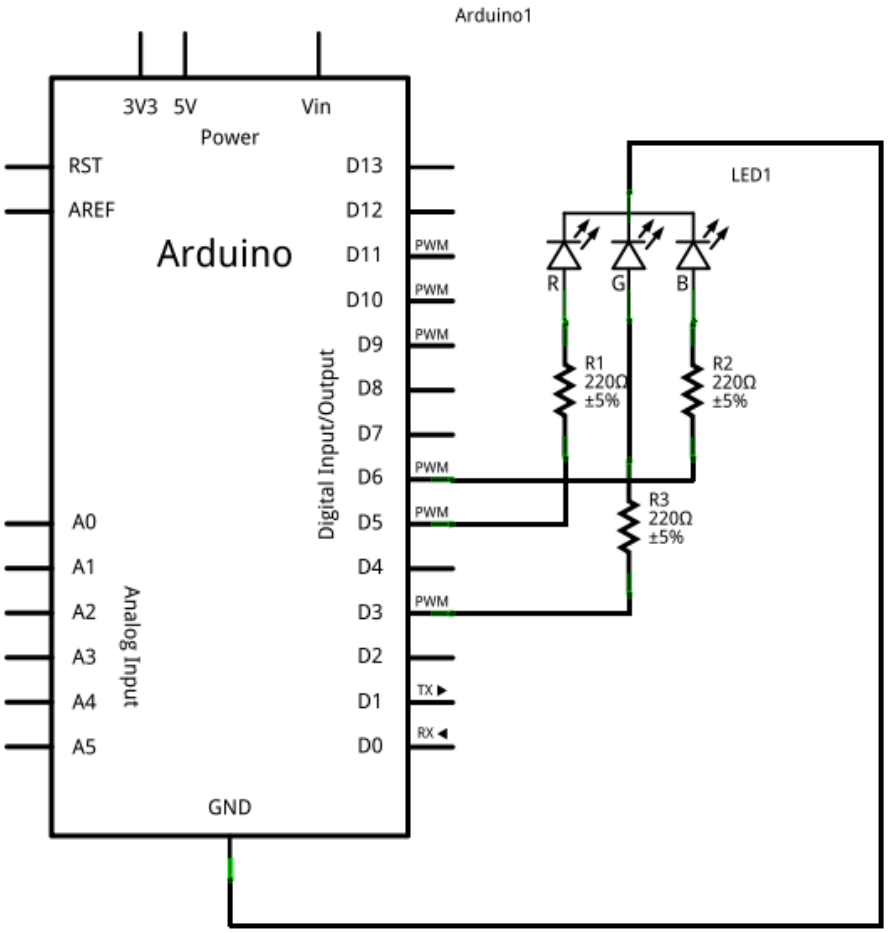

Figura 21: Diagrama Esquemático do node1, sem a representação do módulo de comunicação

5.2. Serviços consumidos pela rede

O único serviço consumido pela rede é realizado no node2. O mesmo, ao saber que é desejo do usuário controlar a cor da lâmpada automaticamente, consome o serviço fornecido pelo node 1.

A Tabela 5 apresenta as informações do único serviço consumido pela rede local. A Figura 22 apresenta a foto do nó node2 da rede.

Tabela 5: Serviço consumido pela rede

Nó local	**Serviço Consumido**	**Método**
node2	corLED	POST

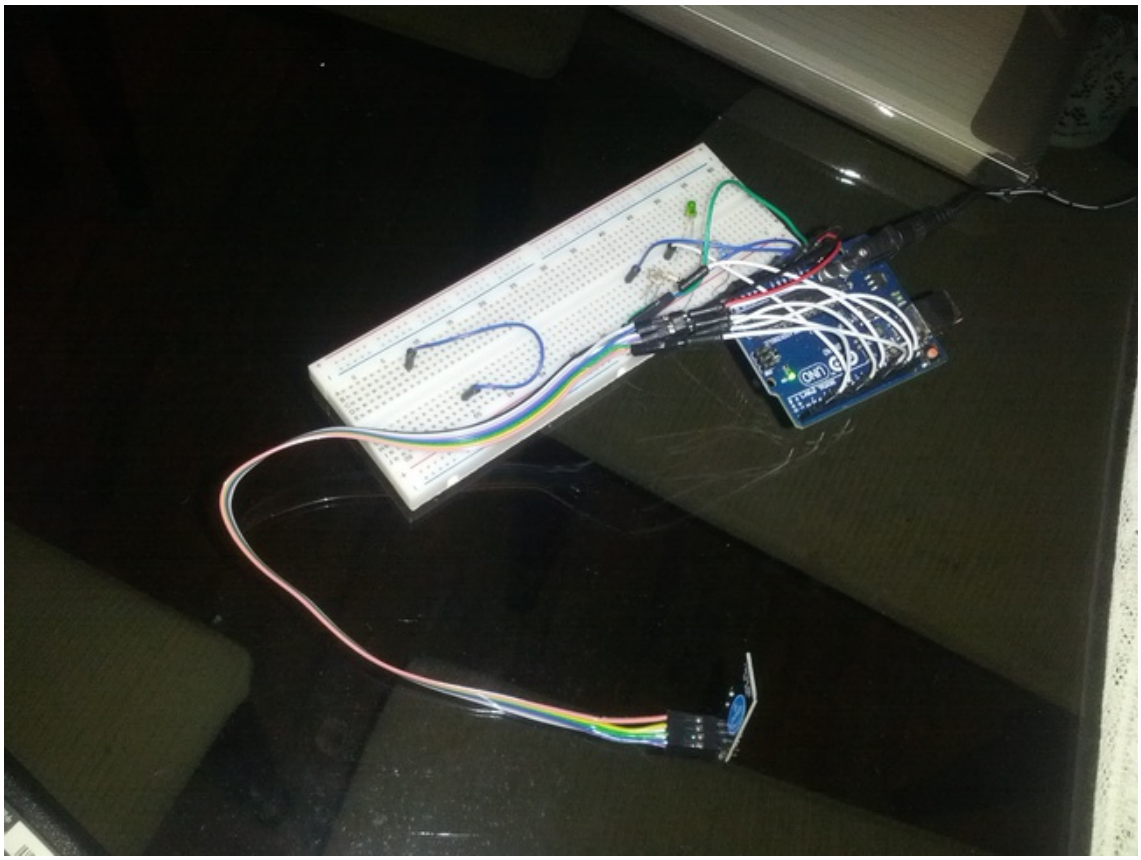

Figura 22: Foto do node2

A Figura 23 apresenta o diagrama esquemático do nó node2.É possível ver que o sensor de luminosidade está conectado a um resistor e à porta A0 do Arduino e que ele possui um LED ligado a ele.

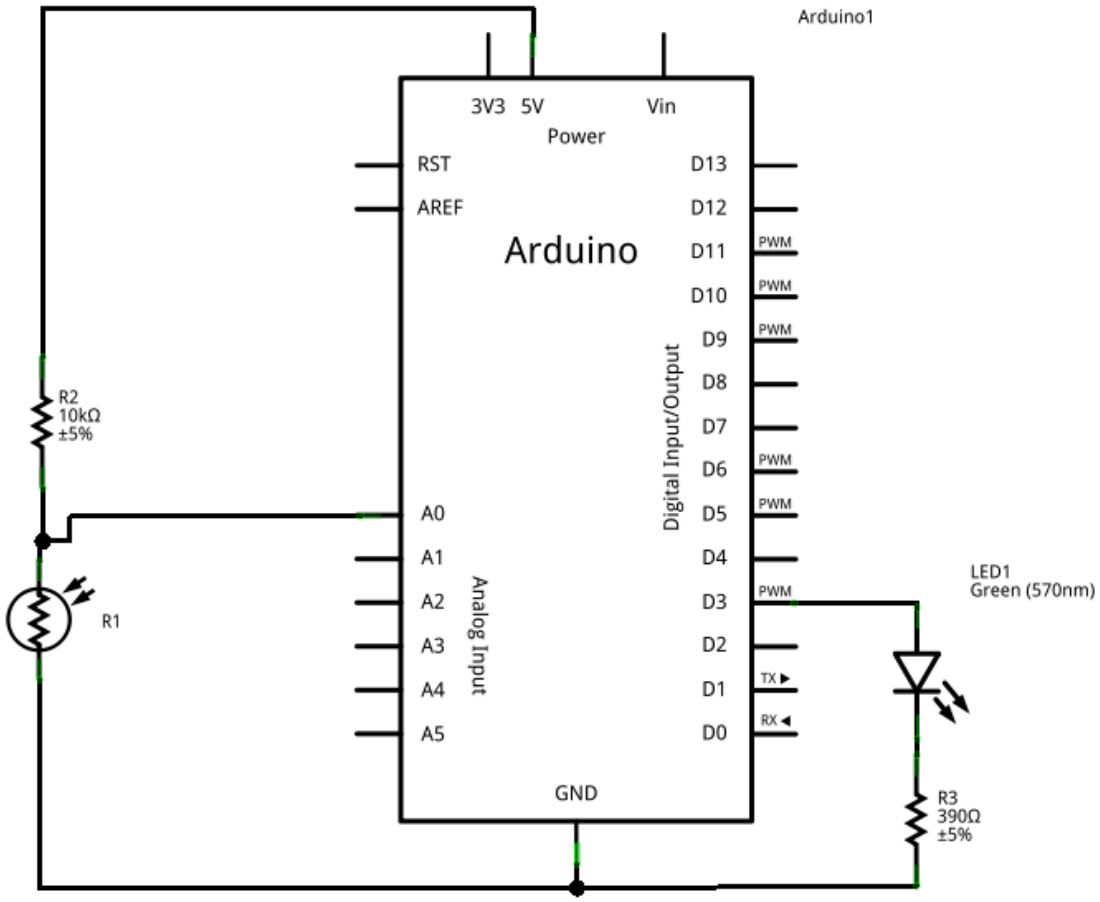

Figura 23: Diagrama Esquemático do node2, sem a representação do módulo de comunicação

5.3. Roteamento no nó Mestre

Pela arquitetura implementada o caso de uso apenas o nó Mestre representa um nó roteador. Ele pode receber pacotes de clientes externos, do node1 e do node2 assim como pode enviar pacotes para os três. Para solucionar o problema de roteamento é realizado uma tomada de decisão a partir do endereço de destino, representado no pacote por addr1.

É necessário fazer a verificação de roteamento tanto no tratamento de um pacote recebido pelo módulo NRF24L01 como pela porta Serial.

A Figura 24 apresenta o diagrama de atividades implementado no nó mestre para permitir o roteamento. Aplicando esse algoritmo é possível controlar o envio dos pacotes da rede local sem precisar enviá-los para o nó de fronteira.

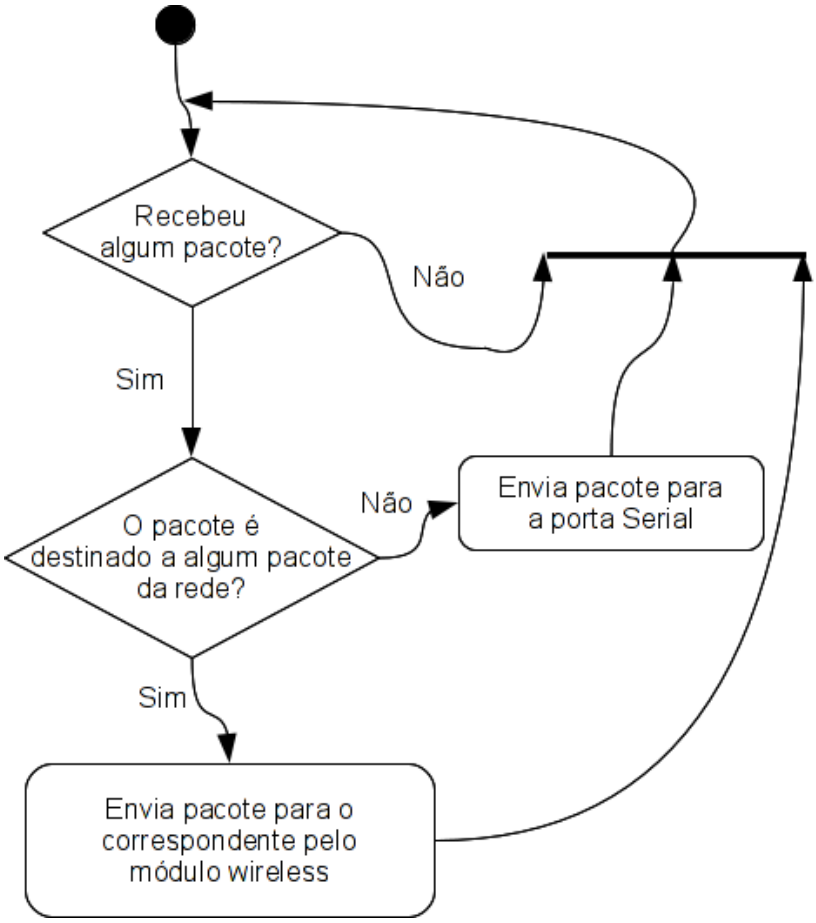

Figura 24: Diagrama de atividates implementado no nó mestre do caso de uso

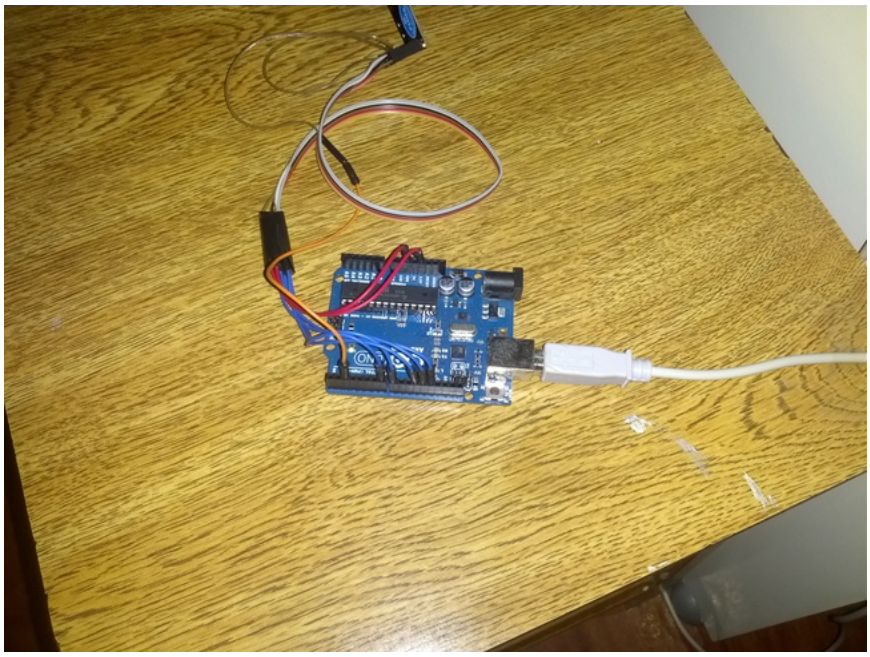

Figura 25: Foto do nó mestre.

A Figura 25 apresenta a foto do nó mestre onde é possível observar a presença apenas das conexões USB e do módulo de comunicação sem fio.

5.4. Adaptação do código do nó local para suportar o SOA

Um requisito do projeto do caso de uso foi fazer a adaptação de um código para que ele possa fornecer e consumir serviços o mais simples possível.

Para se evitar muitas alterações no código implementado no sistema embarcado foi pensado em primeiramente adaptar o comportamento do sistema por valores de variáveis globais.

No caso do node1, que controla a cor da lâmpada RGB, é possível armazenar os valores de cada tonalidade em variáveis globais, cada um armazenando um valor entre 0 e 255. O fluxo principal do programa envia o nível de cada cor, armazenado nas variáveis globais, ao LED RGB.

No caso do node2, uma variável global armazenaria um valor lógico que informaria ao fluxo principal se ele deve ou não consumir os serviços disponibilizados pelo node1.

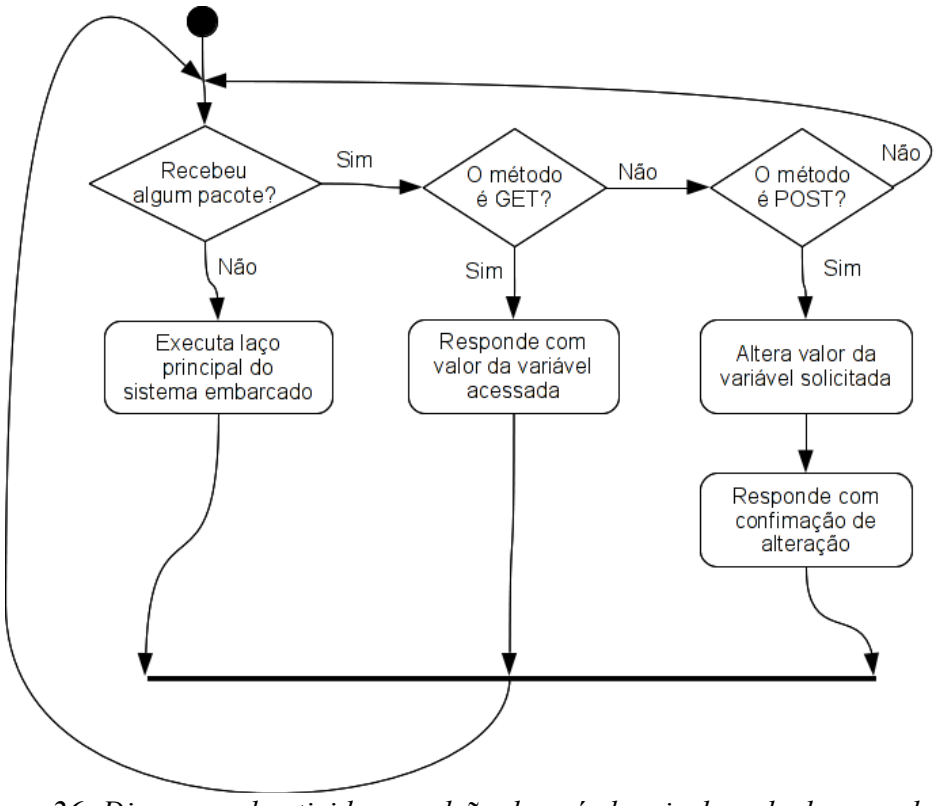

Figura 26: Diagrama de atividaes padrão dos nós locais da rede do caso de uso

A Figura 26 apresenta o diagrama de atividades implementado nos dois nós

locais para o processamento dos pacotes. Da forma que ele foi modelado é possível utilizá-lo em outros tipos de nós locais.

Com o fluxo principal funcionando de acordo com variáveis globais, a implementação da SOA é facilitado. Os serviços disponibilizados podem ser baseados no acesso e na alteração dos valores das variáveis globais. Um bom resultado decorrente dessa abordagem é a simplificação do serviço para alterações e acessos às variáveis do programa.

5.5. Implementação do código do Nó de Fronteira

O nó de fronteira deve ter ciência dos serviços disponibilizados pela rede, na compactação do pacote a mensagem enviada para cada módulo possui informações individualizadas, nesse trabalho não foi implementado uma padronização da mensagem trafegada pela rede local.

Sabendo dos dois serviços disponibilizados na rede, foi escolhido como seriam passadas as informações para o nó local e como as informações da resposta seriam enviadas. Assim foi possível implementar o código para o nó de fronteira.

Primeiramente são adquiridas as informações importantes e independentes de serviço, são elas: Método HTTP e Nó destino. O método é armazenado e guardado no pacote que será transmitido para a rede local e com a informação do Nó destino é guardado no pacote a informação do endereço, no caso do node1 o endereço 1 e no caso do node2 o endereço 2.

Sabendo qual é o nó destino e o método requisitado, verifica-se qual o serviço requisitado daquele nó e captura-se as informações necessárias para aquele serviço. As informações são compactadas da forma que o nó irá compreender e o pacote é enviado para o nó mestre, que por sua vez fará o roteamento até o nó destino. A informação para a resposta é guardada em uma estrutura mapeadora, sendo a chave da mesma a concatenação do endereço destino, endereço origem e o identificador. O identificador é incrementado para o próximo pacote e é assim que requisição de um cliente externo é processada.

O processo da resposta no nó de fronteira é iniciado ao receber o dado pela porta serial. Ao receber o pacote, o mesmo é processado da seguinte forma: Primeiramente recupera-se a informação de resposta armazenada anteriormente, logo após escreve na resposta o código de resposta necessário para o protoclo HTTP, as

informações são descompactadas na forma de um valor que é enviado para o cliente que realizou a requisição.

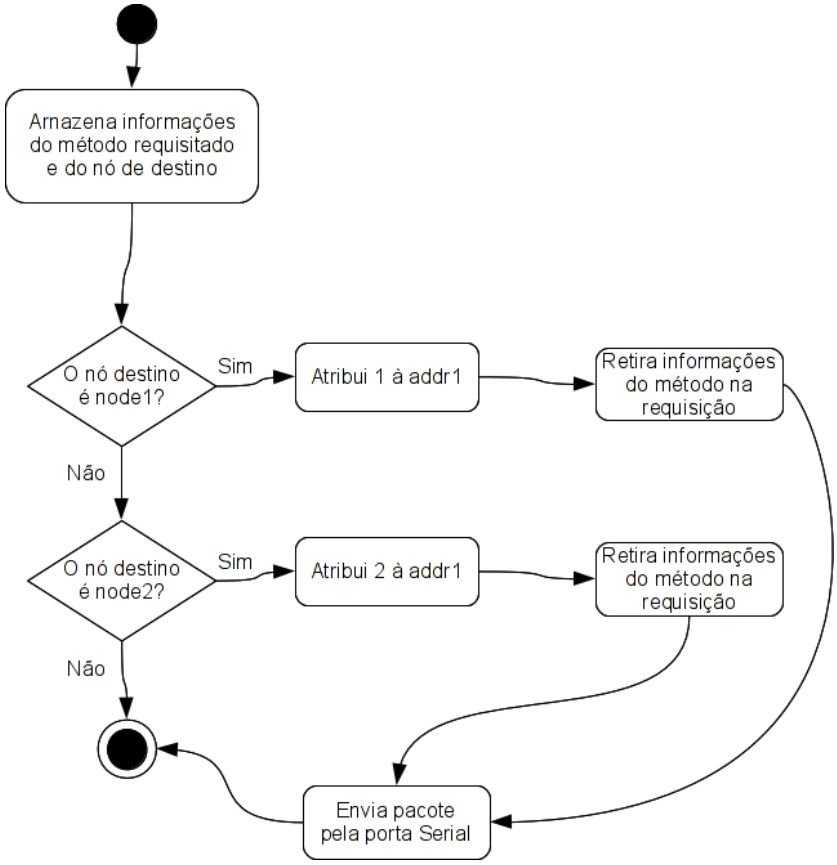

Figura 27: Diagrama de Atividades do processamento da requisição realizado no nó de fronteira

A Figura 27 apresenta o diagrama de atividades presente no no de fronteira para o processamento de uma requisição externa. O nó de fronteira ainda não consegue processar uma requisição de um nó local. É importante observar que esse algoritmo está bastante ligado ao caso de uso sendo necessário uma abstração maior para que ele possa ser aplicado em outras implementações.

A Figura 28 mostra a foto do nó de fronteira do caso de uso e suas conexões. O nó de fronteira está conectado ao nó mestre, ao roteador, a um monitor e à rede elétrica.

Figura 28: Foto do Nó de Fronteira, suas conexões são a Ethernet, o HDMI, o USB de alimentação e o USB do nó mestre

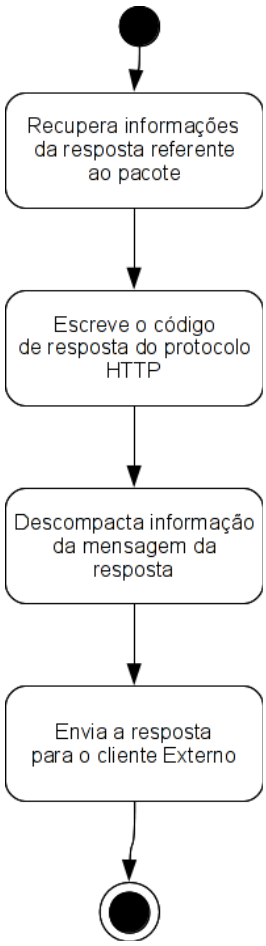

Figura 29: Diagrama de atividades do processamento da resposta realizado no nó de fronteira

A Figura 29 apresenta o diagrama de atividades do processamento da resposta recebida pela rede local. Seu algoritmo está bastante dependente da forma como a requisição é processada, além de não haver uma abstração para o estado de uma requisição externa.

5.6. Cliente Externo

Para o cliente externo foi implementado uma página *web* com três seletores e um botão. Cada seletor representa um dos tons do LED RGB e ao mudar o valor é enviada uma requisição para o node1. O botão ativa ou desativa a funcionalidade automática do node2.

Para a interface gráfica foi usada a biblioteca jQueryUI, o seletor foi implementado com o slider da biblioteca citada, o botão foi implementado com o button da biblioteca citada.

O consumo de serviço pelo cliente externo foi implementado com uma requisição AJAX. É realizado uma requisição HTTP GET ou POST, a depender do serviço, e o servidor a processa, retornando um valor que o cliente saberá processar.

A tela do cliente externo pode ser vista na Figura 30, o cliente foi programado em Javascript. Ele consome os serviços da rede local através de requisições HTTP assíncronas.

Figura 30: Cliente externo escrito em JavaScript

6. Resultados Alcançados

6.1. Resultados Diretos

Dos objetivos pretendidos no início do trabalho, todos foram alcançados com sucesso. Foi feito o projeto dos nós da rede implementando a arquitetura orientada a serviço, os nós foram conectados em uma rede sem fio de baixo custo, a rede local se integrou à Internet através do nó de fronteira e houve a implementação de um caso de uso mostrando como a arquitetura projetada pode funcionar.

O projeto dos nós da rede foi feito levando-se em conta as possibilidades de atuação do mesmo. As possíveis classificações do nó podem auxiliar na compreensão de como um nó deve ser implementado. Por exemplo: Se o nó for um nó roteador, então é necessário fazer uma verificação de endereço no mesmo, caso contrário não é necessário; Se o nó for atuador ou sensor é bastante provável que ele irá fornecer serviços.

Os nós foram conectados em uma rede sem fio de baixo custo, apesar do roteamento ter sido implementao de forma estática, os nós podem conversar de forma não-direta graças aos nós roteadores, além disso é possível acessá-los através da Internet mesmo que os nós não possam se conectar diretamente à mesma.

O nó de fronteira realiza a transcodificação do protocolo HTTP utilizado como condutor de um serviço *web*serviceRESTful permitindo que os nós locais forneçam serviços através da Internet.

O caso de uso planejado foi implementado com sucesso e foi possível verificar alguns padrões que podem auxiliar no desenvolvimento de uma biblioteca para auxiliar o desenvolvimento de módulos que se conectem a essa rede.

De forma geral, o que foi pretendido foi alcançado com sucesso, os resultados foram o esperado e alguns deles surpreenderam, como no caso de uma possível padronização de módulos simples através da modelagem por variáveis globais.

6.2. Resultados Agregados

Os resultados agregados foram aqueles que não foram buscados no começo,

mas que vieram como consequência de outros problemas analisados e resolvidos durante a execução do trabalho, dentre esses resultados é importante citar: Programa para gerar uma sequência de instruções para compactar e descompactar pacotes, uma possível padronização para adaptar certos programas embarcados em uma arquitetura orientada a serviço e uma maior compreensão acerca das dificuldades da Internet das Coisas.

Primeiramente, é possível que no decorrer do trabalho surgiu um problema da codificação e decodificação de um pacote assim como a falta de produtividade ter que reescrever as funções responsáveis por essa tarefa a cada mudança do pacote. É difícil manter uma descrição de pacote inalterada durante o projeto do mesmo visto que durante o desenvolvimento algumas falhas são percebidas e a correção é necessária. Para melhorar a produtividade foi desenvolvido um programa que dada uma descrição de pacotes é gerada uma sequência de instruções que irão compactar uma estrutura em um vetor de bytes e descompactar o vetor em uma estrutura. É possível ver o exemplo de descrição e de código gerado na sessão referenciada.

Outro resultado que foi percebido durante o desenvolvimento destre trabalho foi uma possível padronização quanto à adaptação de um sistema embarcado para uma arquitetura orientada a serviço. Caso seja possível adaptar o fluxo natural do programa parametrizando-o a partir de variáveis globais, os serviços envolvidos no mesmo podem se basear na leitura e na alteração dessas variáveis globais, assim como as funções modificadores de atributos na orientação a objetos. Além do mais o mapeamento desses serviços para um serviço REST torna-se trivial atribuindo os serviços de acesso como métodos GET e os serviços de alteração como métodos POST.

E talvez o resultado mais importante foi a conscientização acerca dos desafitos relacionados a Internet das Coisas, quanto à padronização, comunicação, roteamento e endereçamento, processamento dos nós, construção e projeto dos mesmos. Essa compreensão será descrita nas conclusões.

Conclusões e sugestões para trabalhos futuros

Das conclusões tiradas desse trabalho, é válido citar primeiramente a complexidade e os problemas que envolvem a Internet das Coisas.

A falta de padronização por um lado atrapalha na hora de decidir como implementar determinada solução, mas por outro lado permite que a criatividade gere experimentos interessantes e construtivos para aqueles que percorrerem por esses caminhos.

A rede implementada não está perto do que se espera de uma rede comercial, mas mesmo se apresentando simples no final, surgiram bastantes problemas que requereram a atenção e foco durante um certo tempo. A análise dessas soluções foi bastante construtiva tanto para a construção do conhecimento acerca da Internet das coisas como para o relacionamento entre diversas áreas de conhecimentos aprendidas durante a graduação.

Outra conclusão importante é ver que o conceito de Internet das Coisas realmente pode agregar bastante valor à domótica afinal no mundo altamente conectado, se o usuário possuir uma casa automatizada seria quase um requisito que ele pudesse de conectar a ela de qualquer local que estivesse.

O conceito de arquitetura orientada a serviço pode auxiliar na comunicação entre um cliente externo e algum sistema presente na casa. A arquitetura REST modela o acesso a serviços em recursos, como uma orientação a objetos, sendo assim é bem intuitiva a adaptação para as 'Coisas', já que elas são objetos realmente objetos.

A rede projetada e implementada pode servir como base para futuros trabalhos, afinal mesmo com suas limitações ela é funcional e possui várias características que podem ser melhoradas como uma maior padronização nos campos dos pacotes, no roteamento, no descobrimento de serviços, na segurança etc. Além disso foram usadas ferramentas livres para o desenvolvimento da mesma, sendo possível replicar todo o código implementado sem custo adicionado de *software*.

De forma geral o trabalho foi bastante produtivo auxiliando na fixação do conhecimento aprendido durante todo o curso integrando áreas que se apresentavam de formas distintas e que podem ser relacionadas e também serve como contribuição para futuros trabalhos que desejam usar conceitos abordados nesse.

Apesar deste trabalho apresentar o projeto e implementação de uma rede funcional, é possível observar que ainda falta a implementação de várias características para que ela se torne realmente adaptável. A implementação apresentada ficou muito dependente do caso de uso apresentado.

Uma primeira sugestão é a abstração das características de um nó local para que se possa desenvolver uma biblioteca que forneça suporte ao desenvolvimento de um nó para a rede local. Essa biblioteca deve fornecer as ferramentas necessárias para que o desenvolvedor possa utilizá-la em um projeto pessoal com o mínimo de alterações possíveis.

A segunda sugestão é a adaptação da camada de transporte para que se possa enviar pacotes com tamanho variado, mesmo que seja necessário dividí-lo em pequenos pedaços. A limitação oferecida pelo módulo utilizado nesse projeto é bastante severa dificultando a abstração de respostas dos nós sendo necessário que o nó de fronteira conheça os serviços oferecidos pela rede local.

A terceira sugestão é a implementação de um roteamento automático entre os nós, ou em um primeiro passo uma configuração remota do roteamento. Da forma apresentada no trabalho, o roteamento é estático e se localiza no código do nó, sendo necessária a reprogramação do mesmo para que se obtenha outra forma de roteamento. Ao se fazer a configuração remota é possível implementar um protocolo para o descobrimento de rotas, semelhante ao usado na pilha de protocolos TCP/IP.

A quarta sugestão se refere ao descobrimento de serviços da rede. Além do roteamento estático e da necessidade do nó de fronteira conhecer os serviços da rede, não há mecanismo algum para o cliente conhecer os serviços oferecidos pela rede local.

A quinta sugestão é a organização didática da implementação de um caso de uso para que se possa utilizar esse trabalho na academia. Esse trabalho permite que alunos implementem algoritmos de processamento de pacotes, comunicação entre nós, implementação de um servidor etc. Há bastantes temas possíveis de serem abordados de forma individual quando há uma estrutura completa funcionando. Atividades focadas em uma característica da rede pode auxiliar no aprendizado além de melhorar a própria arquitetura, sendo possível a implementação de alguma sugestão anterior.

Referências

AIELLO, Marco; DUSTDAR, Schahram. Are our homes ready for services? a domotic infrastructure based on the web service stack. **Pervasive and Mobile Computing**, v. 4, n. 4, p. 506-525, 2008.

ATZORI, Luigi; IERA, Antonio; MORA*BIT*O, Giacomo. **The Internet of Things: A survey**. Computer Networks, 54, 1 de Junho de 2010.

AURESIDE. **Casa Inteligente Inclusiva**, 4 de março de 2010. Disponível em <http://www.aureside.org.br/artigos/default.asp?file=01.asp&id=87>, acesso em 20 de março de 2013.

AURESIDE. **Curso de Integrador de Sistemas Residenciais**. Disponível em <http://www.aureside.org.br/treinamento/default.asp?file=introducao.asp>, acesso em 11 de agosto de 2013.

AUTO-ID LABS. Disponível em <http://www.autoidlabs.org>, acesso em 3 de abril de 2013.

BUCKL, Christian et al. Services to the field: An approach for resource constrained sensor/actor networks. In: **Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on**. IEEE, 2009. p. 476-481.

CASTELLANI, Angelo Paolo et al. Architecture and protocols for the internet of things: A case study. In: **Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on**. IEEE, 2010. p. 678-683.

DAY, John D.; ZIMMERMANN, Hubert. The OSI reference model. **Proceedings of the IEEE**, v. 71, n. 12, p. 1334-1340, 1983.

DRISCOLL, Edward B., Jr.. **The history of X10**. Disponível em <http://home.planet.nl/~lhendrix/x10_history.htm>, acesso em 20 de setembro de 2013.

DUNKELS, Adam; VASSEUR, JP. **IP for Smart Objects,** Internet Protocol for Smart Objects (IPSO) Alliance, White Paper #1, 2008.

GERSHENFELD, Neil; KRIKORIAN, Raffi; COHEN, Danny. **The Internet of Things. Scientific America**, p.76-81, 1 de outubro de 2004

GÓMEZ-GOIRI, Aitor et al. Collaboration of sensors and actuators through

triple spaces. In: **Sensors, 2011 IEEE**. IEEE, 2011. p. 651-654.

HUI, Jonathan; CULLER, David; CHAKRABARTI, Samita. **6LoWPAN: Incorporating IEEE 802.15.4 into the IP Architecture,** Internet Protocol for Smart Objects (IPSO) Alliance, White Paper #3, 2009.

IEEE. **IEEE 802.15 WPAN™ Task Group 4 (TG4)**. Disponível em <http://www.ieee802.org/15/pub/TG4.html>, acesso em 3 de abril de 2013.

ILUFLEX. **Iluflex – Automação Sem Fio**. Disponível em <http://www.iluflex.com.br/>, acesso em 2 de outubro de 2012.

KUROSE, James; ROSS, Keith. Redes de Computadores e Internet. **São Paulo: Person**, 2006.

MIORI, Vittorio et al. An open standard solution for domotic interoperability. **Consumer Electronics, IEEE Transactions on**, v. 52, n. 1, p. 97-103, 2006.

PRESSER, Mirko; GLUHAK, Alexander. **The Internet of Things: Connecting the Real World with the Digital World**, EURESCOM mess@ge – The Maganize for Telecom Insiders, vol. 2, 2009. Disponível em <http://archive.eurescom.eu/message/messageSep2009/The-Internet-of-Thing %20-Connecting-the-real-world-with-the-digital-world.asp>, acesso em 3 de abril de 2013.

RFC. **Hypertext Transfer Protocol – HTTP/1.1**. Disponível em <http://www.rfc-editor.org/rfc/rfc2616.txt>, acesso em 16 de agosto de 2013.

SENA, Diane C. S.. **Automação Residencial**. Projeto de Graduação (Graduação em Engenharia Elétrica) – Departamento de Engenharia Elétrica. Espírito Santo: Universidade Federal do Espírito Santo, 2005.

SHELBY, Zach et al. NanoIP: the zen of embedded networking. In: **Communications, 2003. ICC'03. IEEE International Conference on**. IEEE, 2003. p. 1218-1222.

SNELL, James. **Resource-oriented vs. Activity-oriented *Web* services**. Disponível em <http://www.ibm.com/developerworks/*web*services/library/ws-restvsoap/>, acesso em 9 de agosto de 2013.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Distributed systems**. Prentice Hall, 2002.

TANENBAUM, Andrew. S. Redes de Computadores. São Paulo: Ed. 2003.

TOMA, Ioan; SIMPERL, Elena; HENCH, Graham. **A joint roadmap for Semantic technologies and the Internet of Things.**

TOPTENREVIEWS. **2013 Best Home Automation Systems Compared and Reviewed**. Disponível em <http://home-automation-systems-review.toptenreviews.com/>, acesso em 23 de abril de 2013.