CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

UNIDADE ARAXÁ

CARLOS DIAS DA SILVA JÚNIOR

DESENVOLVIMENTO DE PROTÓTIPO DE REDE DE SENSORES SEM FIO

Araxá-MG

2017

CARLOS DIAS DA SILVA JÚNIOR

DESENVOLVIMENTO DE PROTÓTIPO DE REDE DE SENSORES SEM FIO

Trabalho de Conclusão de Curso apresentado ao Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá, como requisito parcial para obtenção de grau de bacharel em Engenharia de Automação Industrial.

Orientador: Prof. Me. Luís Paulo Fagundes

Araxá-MG

2017

Serviço Público Federal
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO CURSO DE ENGENHARIA DE AUTOMAÇÃO INDUSTRIAL / ARAXÁ

TRABALHO DE CONCLUSÃO DE CURSO – ATA DE DEFESA

ATA DA DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DE ENGENHARIA DE AUTOMAÇÃO INDUSTRIAL DO ALUNO **CARLOS DIAS DA SILVA JÚNIOR**

Às 16 horas e 30 minutos do dia 06 de julho de 2017, reuniu-se, no Centro Federal de Educação Tecnológica de Minas Gerais / Unidade Araxá, a Comissão Examinadora de Trabalho de Conclusão de Curso para julgar, em exame final, o trabalho intitulado ***Desenvolvimento de protótipo de rede de sensores sem fio***, como parte dos requisitos para a obtenção do Título de Engenheiro de Automação Industrial. Abrindo a sessão, o Presidente da Comissão, *Prof. Me. Luís Paulo Fagundes,* após dar a conhecer aos presentes o teor das Normas Regulamentares do Trabalho Final, passou a palavra ao candidato para apresentação de seu trabalho. Seguiu-se a arguição pelos examinadores, com a respectiva defesa do candidato. Logo após, a Comissão se reuniu, sem a presença do candidato e do público, para julgamento e expedição do resultado final.

Após a reunião da Banca Avaliadora, o candidato foi considerado Aprovado com nota final de: 88 / 100 (Oitenta e oito pontos em cem pontos).

O resultado final foi comunicado publicamente ao candidato pela Presidente da Comissão. O aluno abaixo-assinado declara que o trabalho ora identificado é de sua autoria material e intelectual, excetuando-se eventuais elementos, tais como passagens de texto, citações, figuras e datas, desde que as mesmas identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos autores, quando necessárias. Declara ainda, neste âmbito, não estar violando direitos de terceiros.

CARLOS DIAS DA SILVA JUNIOR - Carlos Dias da Silva Junior - Araxá, 06 de julho de 2017.
Nome do Aluno | Assinatura | Local e Data

Nada mais havendo a tratar, o Presidente encerrou os trabalhos e lavrou a presente ATA, que será assinada por todos os membros participantes da Comissão Examinadora.

Araxá, 06 de julho de 2017.

Luis Paulo Fagundes
Prof. Me. Luís Paulo Fagundes
Orientador

Leandro Resende Mattioli
Prof. Me. Leandro Resende Mattioli
Avaliador

Kleber Lopes Fontoura
Prof. Dr. Kleber Lopes Fontoura
Avaliador

Prof. Dr. Henrique José Avelar
Avaliador/Suplente

“Não vês que somos viajantes? E tu me perguntas: que é viajar? Eu respondo com uma palavra: é avançar! Experimentais isto em ti: que nunca te satisfaças com aquilo que és para que sejas um dia aquilo que ainda não és. Avança sempre! Não fiques parado no caminho.”

SANTO AGOSTINHO

RESUMO

As Redes de Sensores Sem Fio (RSSF) são caracterizadas por utilizar de comunicação sem fio entre os dispositivos no campo, em vez dos tradicionais cabos de barramento de comunicação, utilizados nas redes industriais comuns. Estas apresentam diversas vantagens e estão sendo utilizadas em diversas aplicações. Este trabalho apresenta um protótipo de rede de sensores wireless que foi desenvolvido com dispositivos de acesso que permitem conectar sensores industriais e um servidor de dados. O protótipo apresentado tentou atender a diferentes demandas de aplicações de RSSF o que não foi possível, pois cada aplicação tem diferentes características. Porem foi possível

Palavras-chave: Rede de sensores sem fio. Sensores industriais. Sistema Supervisório.

ABSTRACT

Wireless Sensor Networks (WSNs) are characterized by wireless communication between devices in the industrial field rather than communications control cables used in common industrial networks. These are presented several times and are being used in many applications. This work presents a prototype of wireless sensor network that was developed with access devices that allow connect industrial sensors into a data server. The prototype of the solution tried to meet different requirements of WSN applications which are not possible, since each application has different characteristics.

Keywords: Wireless sensor network. Industrial sensors. Supervisory system.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CLP - Controlador Lógico Programável;

DMS - Database Management Systems
Sistemas de gerenciamento de banco de dados

GPIO - General Purpose Input/Output
Entrada/Saída de uso geral

HTTP - Hypertext Transfer Protocol
Protocolo de transferência de hypertexto

JSON - JavaScript Object Notation
Notação de Objetos JavaScript

OSI - Open System Interconnection
Sistema de interconexão aberto

REST - Representational State Transfer
Transferência de estado representacional

RSSF - Rede de Sensores Sem Fio

SCADA - Supervisory Control and Data Acquisition
Sistemas de Supervisão e Aquisição de Dados;

SSID - Service Set Identifier
Identificador de conjunto de serviços

SOAP - Simple Object Access Protocol
Protocolo Simples de Acesso a Objetos

SoC - System On a Chip
Sistema em um circuito integrado;

WSN - Wireless Sensor Network
Rede de Sensores Sem Fio

XML - Extensible Markup Language
Linguagem de Marcação Extensível

SUMÁRIO

1 INTRODUÇÃO

As indústrias estão em constante desenvolvimento tecnológico, a fim de acompanhar a concorrência e serem bem-sucedidas. Assim, os fabricantes buscam produzir de forma mais eficiente, com alta qualidade e baixo custo. Para isso, buscam integrar as diversas partes do processo, buscando ter um controle eficaz, através de ferramentas como sistemas SCADA, para a obtenção de dados relativos ao sistema de produção (FERREIRA, 2009).

Neste contexto, as redes industriais são compostas por diversos sensores e atuadores espalhados pelo chão de fábrica e permitem a disponibilização de dados do processo para dispositivos como controladores, historiadores, sistemas de gerenciamento de planta, dentre outros. Essas redes seguem padrões comerciais, criados por diversos fabricantes de componentes para a indústria.

Como uma das opções no campo das redes industriais, temos a Rede de Sensores Sem Fio (RSSF) - Wireless Sensor Network (WSN). Estas redes são caracterizadas por utilizar de comunicação sem fio entre os dispositivos no campo, em vez dos tradicionais cabos de barramento de comunicação.

As redes de sensores sem fio tendem a serem sistemas distribuídos, em que os dispositivos trabalham de forma autônoma (LOUREIRO, 2003). Nesses sistemas, os sensores também podem atuar como controladores, enviando sinais de controle a um atuador, de acordo com o processo.

Conforme Chong et al. (2003) cada nó na RSSF tem capacidade de processamento embarcada e pode ter vários sensores. Os autores exemplificam seu uso no aumento da quantidade de informações através do conhecimento da posição, latitude e longitude, para sistemas móveis.

Além disso, as RSSF proporcionam vantagens de custo e facilidade de implantação, flexibilidade para expansão, economia de energia, entre outras, em comparação com redes industriais com fio comuns. Podem também ser aplicadas em diversos ambientes.

Na fase de implantação, as RSSF não necessitam que cabos cheguem a cada nó, o que reduz os custos, além de necessitar de uma menor quantidade de mão de obra (SILVA; FRUETT, 2010). Caso seja necessária a expansão da rede, basta configurar os novos nós para se conectarem à mesma. No quesito da economia de energia, redes de sensores sem fio não têm quedas de potência no

barramento de comunicação, além de que, na maioria das aplicações, os nós são configurados para consumir o mínimo possível de energia.

Devido a estas diversas características, as RSSF têm um campo de aplicação muito amplo que vai além do ambiente industrial: há pesquisas em serviços de saúde (KO et al., 2015), monitoramento ambiental (SILVA; FRUETT, 2010) e até mesmo operações militares (JANGAM, 2016).

Cada um desses campos de aplicação aproveita de forma diferente os recursos disponíveis. O monitoramento ambiental de Mainwaring et al. (2002), por exemplo, foi feito através de diversos pequenos sensores, alimentados por baterias, que transmitem dados do habitat monitorado, tirando vantagem do fato dos nós da RSSF serem independentes, sem conexões físicas entre si.

Além disso, com a grande expansão da Internet, tornou-se viável disponibilizar dados de diversos sensores. Esta facilidade permite que sensores sejam adicionados a diversos objetos em nosso dia a dia, criando a chamada “Internet das Coisas”.

A “Internet das Coisas” constitui uma das tecnologias mais importantes, com potencial para afetar profundamente nosso modo de vida (ABFMEZIEM, 2014). Basicamente, cada objeto de nossas vidas estará interligado, trocando informações, advindas de diversos sensores, interagindo para alcançar um objetivo comum, e as redes de sensores farão estas conexões.

As aplicações da Internet das coisas e as Redes de Sensores Sem Fio no contexto industrial estão em constante desenvolvimento (MIORANDI, 2012), e podem ser utilizadas para monitorar diversas variáveis, em diversas situações, como sensores externos de gases que podem contaminar a atmosfera, ou monitoramento de dados em áreas de difícil acesso ou perigosas para a presença humana conforme Akyildiz et al. (2002) e Loureiro et al. (2003).

Este trabalho visa contribuir para o desenvolvimento das tecnologias citadas anteriormente, através do desenvolvimento de um protótipo de rede de sensores sem fio.

O presente trabalho tomou como princípios para o desenvolvimento do protótipo de rede de sensores sem fio:

- Nós de sensoriamento de baixo custo e compactos;
- Arquitetura escalável, ou seja, a RSSF suporta a adição de novos nós;

- Melhor gerenciamento dos recursos de processamento e memória, seja nos sistemas embarcados, seja no servidor de dados;
- Integração com a internet e outros sistemas.

2 REFERENCIAL TEÓRICO

Para projetar uma RSSF, deve-se levar em consideração a tolerância a falhas, a escalabilidade, custos de produção, ambiente operacional, topologia da rede, restrições de hardware, meio de transmissão e consumo de energia (RUIZ, 2004). Alguns destes pontos são tratados por Gungor (2009) como “desafios” no projeto de uma RSSF. Todos estes itens são de grande importância, pois servem como pilares para definições de *hardware*, topologia da rede, algoritmos embarcados, etc.

Por exemplo, Ko et al. (2010) tiveram grande preocupação com alguns pontos relevantes à sua aplicação, área da saúde, como a necessidade de alta confiabilidade nos sensores, ou seja, a necessidade de o sistema conseguir realizar a entrega de dados sem perdas de pacotes. A segurança e privacidade dos dados foram outras questões que os autores consideraram importante devido ao conteúdo destes serem sigilosos e particulares para os pacientes.

Já Silva e Fruett (2010), devido à natureza da aplicação de seu sistema - monitoramento ambiental - tinham como prioridades o baixo consumo de energia, devido a não existência de uma fonte de eletricidade convencional no local, e possibilidade de gerenciar e monitorar os sensores a distância.

Neste capítulo, são descritas as decisões que foram tomadas com base em trabalhos de outros autores, além de algumas definições que foram necessárias para o desenvolvimento deste trabalho. Inicia-se com o hardware dos nós - que impacta diretamente nos custos, tipo de arquitetura e escalabilidade - até a forma de integração com outros sistemas.

2.1 Hardware dos nós

Conforme RUIZ (2004), uma RSSF é formada por diversos dispositivos autônomos que geralmente são projetados para serem compactos Estes, chamados “nós da rede”, têm como principais componentes um módulo de comunicação sem fio, fonte de energia, um sensor e um microcontrolador.

Na aplicação de redes de sensores sem fio para a saúde, Ko et al. (2010) utilizaram uma plataforma comercial chamada Shimmer. Esta plataforma nos oferece com baixo consumo de energia e recursos voltados para as áreas de saúde.

Estes são constituídos por um microcontrolador de 16 bits da Texas Instruments, o MSP430, e um sistema de rádio de baixa potência, Chipcon CC2420.

A opção de unidade de sensoriamento com as funções mais básicas é um dispositivo com os recursos necessários para acesso a rede sem fio além de um acelerômetro, giroscópio, magnetômetro e um cartão de memória embarcado, como o da FIG. 1.

Figura 1 - Placa Shimmer.
Fonte: Ko et al. (2010, p. 6).

O custo de uma unidade básica é de pouco mais de R$1300[1], conforme o site da plataforma (www.shimmersensing.com). Do ponto de vista dos componentes utilizados nos nós do Shimmer, um kit de desenvolvimento básico para o microcontrolador TI MSP430 varia entre R$40 e R$60 - de acordo com as funcionalidades oferecidas pelo kit, enquanto um módulo de rádio com o Chipcon CC2420 custa R$40. É importante salientar que há valores agregados na produção do Shimmer que não foram levantados, como custos de produção de placas e desenvolvimento dos softwares envolvidos.

Em seu trabalho de monitoramento ambiental, Silva e Fruett (2010) utilizaram de um microcontrolador de 32 bits da NXP com o núcleo de processamento ARM Cortex-M0.

Conforme explicado pelos autores, Utilizar esta família de microcontroladores tem como vantagens um conjunto de instruções aprimorado, aumentar a autonomia de energia e apresentar uma capacidade de processamento considerável. Para conectar este processador à rede sem fio, os autores optaram

1 Os valores de preço apresentados estão de acordo com pesquisa realizada no dia 28 de junho de 2017 e foram aproximados sem levar em consideração impostos.

por utilizar um módulo que já é de utilização comercial, o X-Bee Pro que utiliza o padrão ZigBee.

Segundo Gungor (2009), ZigBee é um padrão de rede de malha com base no padrão IEEE 802.15.4 que é voltado para o controle e monitoramento industrial, automação residencial, entre outros. Conforme o autor, ZigBee é promovido por um grande consórcio de empresas e tem como principais vantagens “consumo de energia extremamente baixo” e suporte a várias topologias diferentes.

Silva e Fruett (2010) argumentam que com este módulo é possível que seja feita uma “comunicação simples entre nós”. O módulo se comunica com o microcontrolador por meio de uma interface UART. O preço médio de um módulo X-bee Pro de 60mW é de R$120.

Neste trabalho, foi escolhido o SoC ESP8266. Um SoC é um sistema completamente embarcado em um único chip. Estes sistemas geralmente incluem microcontroladores integrados com sensores, memória adicional para armazenamento de dados, hardware para comunicação sem fio, etc.

No caso do SoC ESP8266, conforme a fabricante deste a Espressif, este contém um microcontrolador Tensilica L106 de 32 bits, que pode atingir um *clock* máximo de 160 MHz. Este é responsável por disponibilizar interfaces de entrada e saída, como portas digitais, comunicação SPI e I2C. No mesmo chip há integrado um hardware de comunicação Wi-Fi IEEE 802.11 b/g/n.

O microcontrolador embarcado permite que sejam desenvolvidos softwares que serão executados no mesmo, os chamados *firmwares.* Para o desenvolvimento destes é possível utilizar de diferentes linguagens de programação como Lua, Python e C++.

O ESP8266 é distribuído na forma de módulos, que são distribuídos e desenvolvidos por fornecedores de eletrônicos. Cada um dos módulos disponibiliza as funcionalidades do chip de formas e a custos diferentes, que podem ser utilizadas de acordo com os requisitos do projeto.

Há, por exemplo, módulos que apresentam apenas uma interface serial e duas portas digitais, que podem ser usadas como entrada e saída de dados. É comum nestes módulos que o *firmware* que recebe comandos de controle do Wi-Fi via serial, sendo então utilizado como um adaptador Wi-Fi para aplicações já desenvolvidas em outras plataformas.

Um dos módulos mais comuns e de menor custo é o ESP-12E que disponibiliza as 11 portas de GPIO do ESP8266, como conversor analógico-digital, comunicação I2C e SPI, entre outros, por um preço médio de R$10,00.

Este módulo tem como limitação a necessidade de se utilizar um conversor USB/UART, para escrito o *firmware* no ESP8266. Outra característica que dificulta seu uso é o fato de que as portas de GPIO estão dispostas de uma forma que impossibilita a sua fixação em uma *protoboard* para testes, necessitando uma adaptação para fazê-lo.

Como uma opção mais completa, há o módulo NodeMCU, da AIThinker, FIG. 2. Este módulo é composto por um chip conversor USB/UART, uma fonte de alimentação regulada de 3.3V e o módulo ESP12E. O custo desse módulo é, em média, R$15,00.

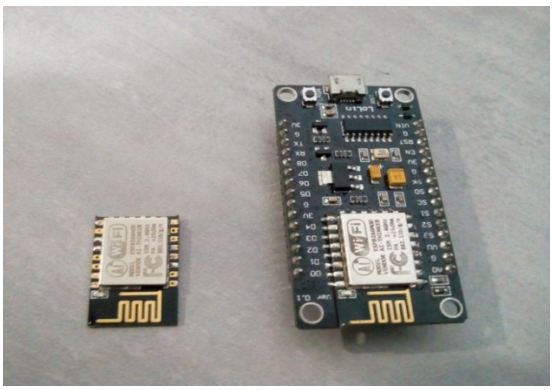

Figura 2 - ESP12-E, à esquerda, e NodeMCU, à direita.

Devido ao preço e ao tamanho da placa do NodeMCU serem maiores que o do ESp13-E somente, pode-se utilizar o módulo NodeMCU durante o período de desenvolvimento e, depois de concluir o software e testar os componentes externos necessários, para produção utilizar o módulo ESP-12E.

2.2 Dados centralizados

Conforme a definição de Kortuem (2010), as redes são compostas por objetos físicos - o hardware, e objetos digitais - os softwares, sendo que ambos devem ser focados na interação com o ser humano.

Também no trabalho de Bröring et al. (2011), um dos pontos chave das redes de sensores sem fio é a necessidade de criação de tecnologias que façam os recursos acessíveis a outras aplicações. Os autores argumentam que devido a

grande variedade de tipos e fabricantes de sensores, com seus próprios protocolos de comunicação, a integração com diferentes sistemas de monitoramento desses dados não é direta.

Para Bröring et al. (2011) é clara a diversidade de formas de disponibilizar os dados dos sensores, de forma descentralizada ou centralizada. Na forma descentralizada, as informações dos dados dos sensores são entregues pelo próprio nó da rede ao cliente, enquanto na centralizada, os dados são enviados a uma central pela qual o cliente pode acessar estes dados.

Na forma descentralizada, um dos exemplos citados pelos autores é uma aplicação em que os dados de cada um dos sensores são enviados a diversos clientes que necessitam destes. Assim, cada cliente recebe somente o que lhe é útil, diretamente da fonte.

Esta forma é limitada pelo número de clientes que podem receber dados de cada nó simultaneamente. Esta limitação é gerada pelo poder de processamento do *hardware* em cada nó, que poderá atender até certa quantidade de clientes, sem gerar atrasos na entrega dos pacotes.

Quanto a sistemas centralizados, os autores tratam da importância dos chamados “Centralized Sensor Web Portals” ou, em tradução livre, os “Portais de redes de sensores centralizados”.

Como exemplo de “Portais web centralizados”, Bröring et al. (2011) citam o portal Pachube, adquirido pela LogMeIn e então renomeado para Xively. Este tem como objetivo fornecer uma plataforma para receber os dados de diversos sensores através da internet, permitindo ao usuário gerenciá-los e monitorá-los em uma interface em tempo real.

No projeto apresentado por Mainwaring (2002) os dados dos sensores são enviados a um servidor no ambiente da aplicação. Este servidor atualiza outro enviando pacotes maiores, com dados de todos os sensores, a cada cinco minutos, devido a limitações de conexão com a internet no local da RSSF. Desta forma os dados da rede podem ser acessados por meio de uma conexão com a Internet, por diversos usuários, mesmo com um pequeno atraso.

Como uma opção para criar uma camada de acesso aos dados há a tecnologia de *Web Service*. Este é um sistema que cria uma interface entre diferentes dispositivos que estão em uma rede, geralmente através do protocolo HTTP.

Conforme a FIG. 3, os clientes devem fazer solicitações de dados ao servidor através do *Web Service*. As solicitações resultam em dados que são entregues pelo servidor ao cliente, também por meio desta interface.

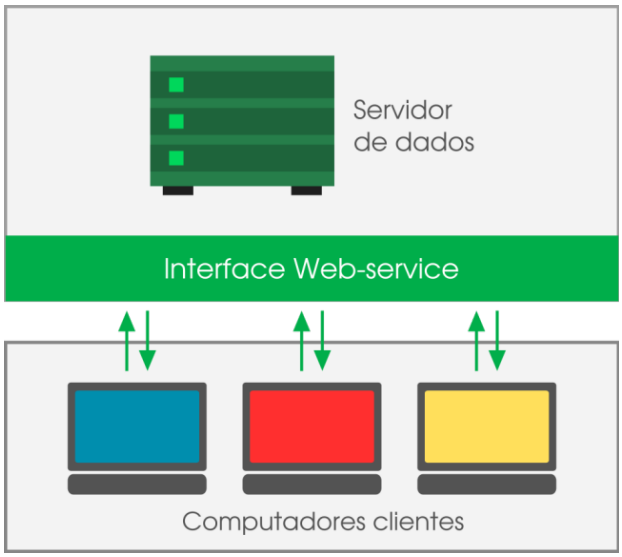

Figura 3 - Comunicação entre clientes e servidor através do Web Service.

Conforme Wagh e Thool (2012) *Web Service* são *softwares* auto descritivos e modulares que expõem dados como serviços na Internet através de uma interface programável e onde um endereço específico é usado para buscar e salvar esses dados. O formato do *Web service* é definido pelo desenvolvedor, dependendo dos objetivos e características esperadas no projeto. Os tipos mais comuns são o Simple Object Access Protocol (SOAP) e Representational State Transfer (*REST*).

O SOAP estabelece regras rígidas de trocas de mensagens através de tecnologias da Extensible Markup Language (XML). Conforme a definição da W3C a estrutura do SOAP foi projetada para ser independente de qualquer modelo de programação particular e outras semânticas específicas de implementação.

Já o REST é um tipo de *Web Service* mais flexível, pois não estabelece restrições ao formato das mensagens. Assim a definição do corpo de dados é definida apenas pelo desenvolvedor, o que pode gerar os problemas para sua utilização.

Toda a comunicação entre cliente e servidor no REST é realizada através de mensagens pelo protocolo HTTP. Conforme é definido pela W3C, o HTTP é um protocolo que se encontra no nível da camada de aplicação no modelo Open System

Interconnection (OSI) e destina-se a sistemas de informação distribuídos e colaborativos e é usado na rede global de computadores desde 1990.

Este protocolo estabelece padrões como sintaxe e requisitos para análise das mensagens, através de oito métodos: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS e CONNECT.

Os métodos podem ser utilizados de acordo com a aplicação, sendo escolhidos pelo desenvolvedor. Muitas vezes, em *web-services*, são utilizados os métodos GET para adquirir informações e o POST para enviar.

O GET adquire e envia informações através do endereço - a URL - da requisição, a qual retorna dados, geralmente em formato de XML ou JavaScript Object Notation (JSON).

Apesar do nome, o JSON não é de uso exclusivo para utilização com JavaScript, podendo ser utilizado em qualquer aplicação. É geralmente aplicado em trocas de mensagens entre servidor e cliente.

O JSON é uma maneira de organizar dados de forma estruturada, que facilita a leitura dos dados por usuário. Seguindo o exemplo de uma agenda de contatos, os dados de uma pessoa poderiam ser armazenados em forma de um JSON como na FIG. 4.

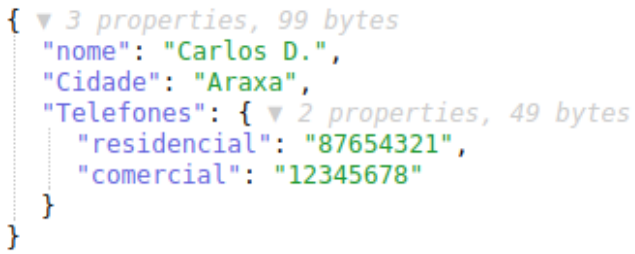

Figura 4: Exemplo de JSON de um contato de uma agenda.

Neste exemplo, é possível identificar três propriedades do contato: o nome, cidade e telefones. Este último ainda é dividido em outras duas propriedades: residencial e comercial.

O JQuery é uma biblioteca de funções em JavaScript como animações e manipulação de eventos e de partes de um documento HTML. Já o Bootstrap é um *framework* HTML, CSS e JS que disponibiliza diversos componentes reutilizáveis, configurações de CSS para diversos elementos do HTML e alguns componentes em

JQuery. Conforme o site do projeto (getbootstrap.com) o Bootstrap acelera e facilita o processo de desenvolvimento de *frontend* para a web.

Já o POST é usado para enviar dados ao servidor, geralmente que alteram o estado de algum recurso. Por exemplo, neste trabalho, para se alterar o nome de algum dos sensores, utiliza-se uma mensagem enviada através do método POST ao servidor.

2.3 Bancos de dados

Basicamente, bancos de dados são conjuntos de dados estruturados. Uma agenda de contatos, por exemplo, é um banco de dados com informações de contato como nome, e-mail e telefone.

Quando se trata de bancos de dados em computadores os dados são organizados por *softwares* chamados de Database Management Systems (DMS) - Sistema de Gerenciamento de Banco de Dados. Estes softwares são responsáveis por permitir a busca e manipulação de dados no banco de dados.

No exemplo da agenda os dados podem estar organizados de diferentes maneiras. É possível, por exemplo, guardar os contatos em ordem alfabética dos nomes ou simplesmente na ordem em que foram adicionados.

É importante observar que independente da forma em que foram armazenados, estes dados sempre irão manter as relações entre eles, de forma que sempre que houver uma busca por um contato, o resultado de seus dados será sempre o mesmo.

Nos DMSs relacionais estes dados são armazenados em uma ou mais tabelas que têm colunas. Assim, a tabela “contatos” tem as colunas nome, e-mail e telefone. As colunas nas tabelas são os conjuntos de dados de mesmo tipo e que têm relação com outros que estão em uma mesma linha. Cada linha desta tabela representa um contato, conforme a FIG. 5**.**

Figura 5: Banco de dados de uma agenda.

No trabalho de Mainwaring (2002) os dados são guardados em um banco de dados PostgreSQL o qual constitui um sistema de gerenciamento de banco de dados relacional de código aberto. Conforme o site do projeto (www.postgresql.org) possui mais de 15 anos de desenvolvimento ativo e pode ser executado nos principais sistemas operacionais, como por exemplo, sistemas GNU/Linux e Windows.

São opções de restrições do campo:

- Não nulo: o valor do campo não pode ser nulo;
- PK: chave primária, ou seja, um número de identificação para a linha inserida;
- *Auto-Increment*: o valor é incrementado automaticamente, a medida que novas linhas são inseridas na tabela;
- *Unique*: único, ou seja, não permite que haja duas linhas com dados neste campo iguais;

Já o *SQLite* não há arquivos de configuração: todo o banco de dados está contido em um único arquivo. Isto facilita, por exemplo, a migração entre servidores, bastando copiar o arquivo para o novo local. O mesmo é útil em situações como na necessidade de se realizar uma cópia de segurança ou compartilhar o banco de dados.

3 MATERIAIS E MÉTODOS

3.1 Algoritmo básico dos nós

Conforme a FIG. 6, o algoritmo dos módulos dos nós, inicia-se quando estes forem ligados a uma fonte de energia. Quando isto ocorre, primeiramente o módulo irá garantir sua conexão com a rede. Para isto, ele deve se conectar a rede especifica da aplicação.

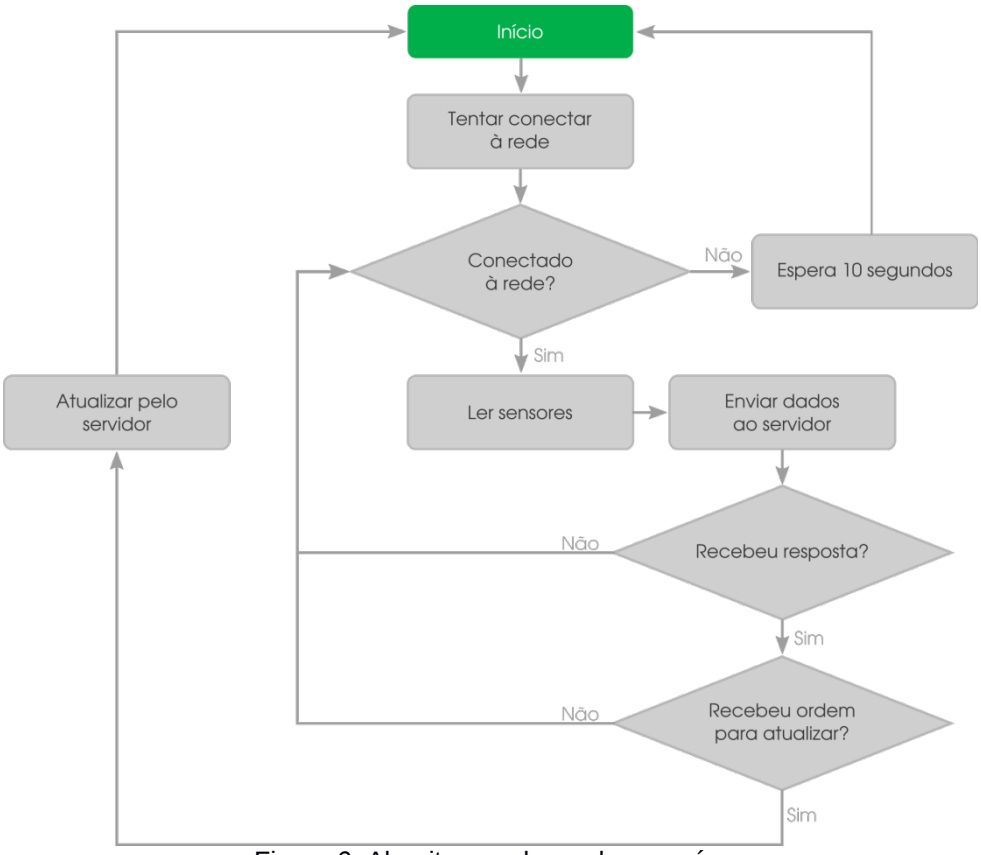

Figura 6: Algoritmo embarcado nos nós.

Os valores do SSID e senha da rede aos quais os nós vão se conectar são fixos, e podem ser alterados apenas com novas versões do firmware. É possível em alterações no software, adicionar várias redes possíveis de conexão.

A plataforma de desenvolvimento do ESP8266 disponibiliza funções para controle do módulo de Wi-Fi. Para se conectar à rede desejada, utiliza-se da função “WiFi.begin”, disponível através da biblioteca “ESP8266WiFi”, cujos parâmetros são o SSID e a senha.

Caso o nó não consiga se conectar à rede, como por exemplo se a rede estiver fora de alcance, o nó espera dez segundos antes de tentar se reconectar.

As entradas de sensores são então lidas, e preparadas para envio. Além dos dados dos sensores, é enviada também a versão do *firmware* que está sendo utilizada, além do número de identificação de *hardware*. Este número é dado pela função “ESP.getChipId()” e corresponde aos três últimos bytes do MAC do ESP8266.

Geralmente para atualizar o *firmware* de um microcontrolador é necessária uma conexão física, através da interface serial, além de, no caso do ESP8266, também iniciá-lo em um modo específico para programação da flash.

Porém, o microcontrolador utilizado tem uma funcionalidade chamada de “Over the Air” (OTA) que permite carregar o firmware no módulo usando conexão Wi-Fi. Este processo pode ser feito de três maneiras: através de uma página de configuração no próprio ESP8266, por meio de uma IDE como a *Arduino IDE* ou através do download do novo firmware de um servidor configurado anteriormente, método que foi utilizado por este trabalho.

Para enviar uma mensagem HTTP através do método POST no ESP8266, utilizou-se a biblioteca “`ESP8266HTTPClient”.` Após enviar a mensagem ao servidor, o ESP8266 aguarda um retorno com a confirmação de recebimento pelo servidor. A mensagem pode ter dois códigos de resposta: “Sem conteúdo” ou “OK”.

Caso o código de resposta seja 204, “Sem conteúdo”, o servidor processou a requisição com sucesso e não há nenhuma atualização disponível para o dispositivo em questão.

Um código de resposta 200, “OK”, significa que a mensagem também foi processada, porém há uma atualização disponível. Neste caso, o nó adquire o arquivo binário com a nova atualização, através do link disponível no corpo da mensagem de retorno.

Caso a mensagem de confirmação não for recebida, e antes de enviar outros dados ao servidor sempre é verificada a conexão com a rede.

3.2 Circuito adaptador para os sensores

Neste trabalho, optou-se por trabalhar com dois tipos de sensores: sensores analógicos na faixa de 0-12 V e digitais de até 30 V. Como estas faixas de valores poderiam causar danos às portas do ESP8266, são necessários circuitos para adaptá-las à uma faixa tolerável.

O conversor analógico-digital do ESP8266 trabalha na faixa de 0-1 V, com resolução de 10 bits. A porta analógica suporta até a tensão nominal do SoC, ou seja 3.3 V, porém valores acima de 1 V são considerados como o máximo valor possível, ou seja 1024.

Para suportar a entrada 0-10 v foi utilizado um divisor de tensão simples, com dois resistores, um de 100KΩ e outro de 10KΩ, conforme a FIG. 7.

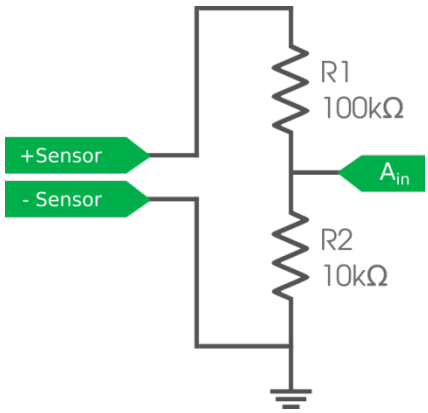

Figura 7: Circuito divisor de tensão para a entrada analógica.

Quando há uma tensão entre os terminais de entrada do sensor igual a *12 V,* o circuito série dos resistores RA1 e RA2 serão percorridos por uma corrente *Itotal = 109 μA,* conforme Eq. 1a. Assim a tensão em *RA2* será de 1,09v, conforme Eq. 1b.

Equação 1: Cálculos para o valor de VRA2

$$I_{total} = \frac{V_{entrada}}{R_{A1} + R_{A2}} \Rightarrow I_{total} = \frac{12}{100\,k + 10\,k} \Rightarrow I_{total} = 109\,\mu\,A$$

$$V_{RA2} = R_{A2} \cdot I_{total} \Rightarrow V_{RA2} = 10\,k \cdot 109\,\mu \Rightarrow V_{RA2} = 1{,}09\,v$$

Sendo assim, mesmo com um valor de entrada 20% acima do máximo esperado a tensão da entrada no ESP8266 ainda é suportada. Isto diminui a

possibilidade de danificar o processador devido a algum pico de tensão no sinal dos sensores.

A tensão em R2, quando a tensão de entrada é de 10v será de 0,91v, , conforme Eq. 2b. Este valor de tensão, o conversor digital-analógico do ESP8266 irá representar na forma do valor decimal de 931. Esta informação é importante para que se obtenha o valor real medido.

Equação 2: Cálculo da tensão em RA2 quando a entrada é de 10v

$$I_{total} = \frac{V_{entrada}}{R_{A1} + R_{A2}} \Rightarrow I_{total} = \frac{10}{100\,k + 10\,k} \Rightarrow I_{total} = 90{,}91\,\mu\,A$$

$$V_{RA2} = R_{A2} \cdot I_{total} \Rightarrow V_{RA2} = 10\,k \cdot 90{,}91\,\mu \Rightarrow V_{RA2} = 0{,}91\,v$$

Já para suportar a entrada de sinais digitais, utilizou-se de um circuito com um transistor NPN, MMBT2222, conforme a FIG. 8. Quando não há tensão nos terminais de entrada do sensor, ou esta é zero, o valor medido em “DIN” é de 3.3v. Quando há tensão, o transistor fica saturado e “DIN” será conectado ao 0v.

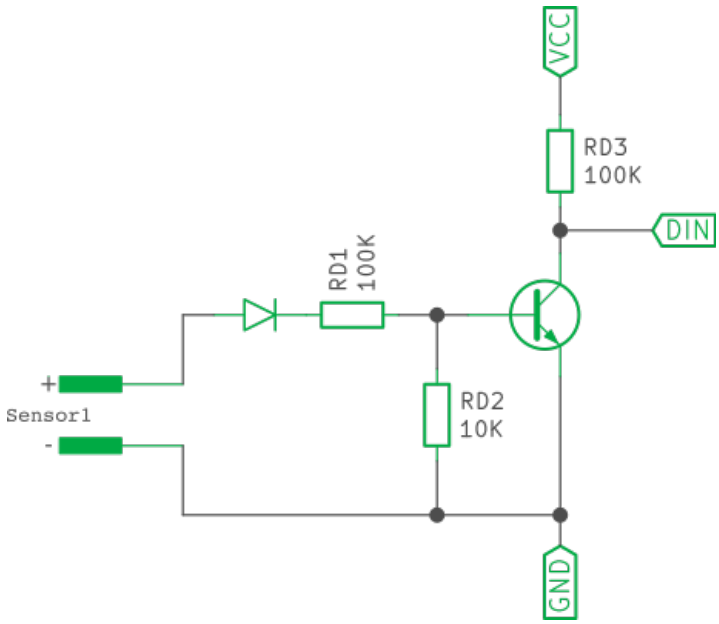

Figura 8: Circuito de um transistor como chave para a entrada digital.

Assim, o valor de “DIN” é sempre o inverso da entrada. A correção é realizada através do *firmware* embarcado no nó.

3.3 Circuito esquemático para o ESP8266

As placas de circuito impresso dos nós devem conter os conversores de sinal analógico e digital, além de um regulador de tensão e as ligações necessárias para comunicação via serial e I2C.

Por se tratar de um SoC, o ESP8266 dispõe de alguns componentes básicos para seu funcionamento, como um cristal para gerar os pulsos de *clock* para o processador e memória flash adicional para armazenamento de dados.

Porém, para que este funcione normalmente é necessário que os pinos de *reset* (REST) e de desligar o chip (CH_PD) estejam em nível lógico alto. Cumprido este requisito, podem-se selecionar os modos de boot do ESP8266 conforme a TAB. 1.

Tabela 1: Modos de boot no ESP8266

GPIO15	GPIO0	GPIO2	Descrição
Nível Baixo	Nível Alto	Nível Alto	Boot normal pela Flash interna
Nível Baixo	Nível Baixo	Nível Alto	Boot em modo programação serial
Nível Alto	Não importa	Não importa	Boot por cartão SD externo

Assim, apesar da possibilidade de se enviar um novo *firmware* via rede sem fio, foi adicionado na placa o circuito necessário para realizar a gravação por meio da comunicação serial.

Esta opção deu-se ao fato que é necessário gravar um código básico, assim que o ESP8266 chega de fábrica, para se conectar à rede sem fio e buscar o novo *firmware*. O modo de gravação da memória *flash* é acionado fechado o contato entre os pinos descritos como “FlashPins”, visíveis na FIG. 11. Estes pinos, quando fechados, fazem com que a GPIO0 esteja em nível lógico baixo, permitindo que o ESP8266 entre no modo de programação via serial.

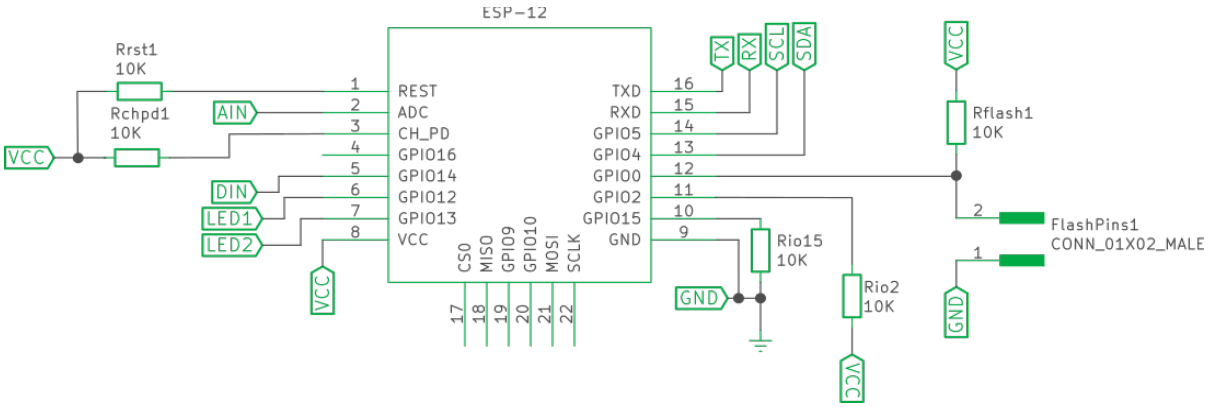

Figura 9: Circuito das conexões do ESP8266.

A porta I2C do ESP8266 foi disponibilizada na placa como um barramento opcional. Este protocolo de comunicação é utilizado por alguns módulos de sensores como acelerômetros, giroscópios e umidade, que podem ser úteis para determinadas aplicações.

Como regulador de tensão, optou-se pelo circuito integrado LM117. Conforme a fabricante do componente, a Texas Instruments, este regulador consegue fornecer até 800mA de corrente de saída, oferecendo proteção para limitação de corrente e contra aquecimento excessivo. Apesar de este regulador estar disponível também em uma versão com a tensão de saída regulável, neste trabalho foi utilizada a versão com valor fixo de 3,3v.

O circuito para utilização deste regulador de tensão é composto apenas por dois capacitores: um em paralelo com a entrada não regulada e outro, também em paralelo, na saída.

Como forma de proteção adicional, há diodos em série na entrada digital. Este diodo impede que seja conectado um sensor com a polaridade do sinal invertida, que poderia danificar o circuito.

Além disso, foram adicionados dois LED's conectados nas portas 12 e 13 do ESP8266. O objetivo da presença deste na placa é ter indicadores visuais para diagnóstico mais rápido de problemas, como falha na conexão com a rede.

3.4 Confecção da placa de circuito impresso

Para o desenvolvimento da placa de circuito impresso, foi utilizado o *software* código aberto KiCad, conforme a FIG.12. Este *software* disponibiliza

diversas ferramentas como um editor de esquemas elétricos, editor de placas de circuito impresso, gerenciador de lista de materiais para fabricação, entre outras.

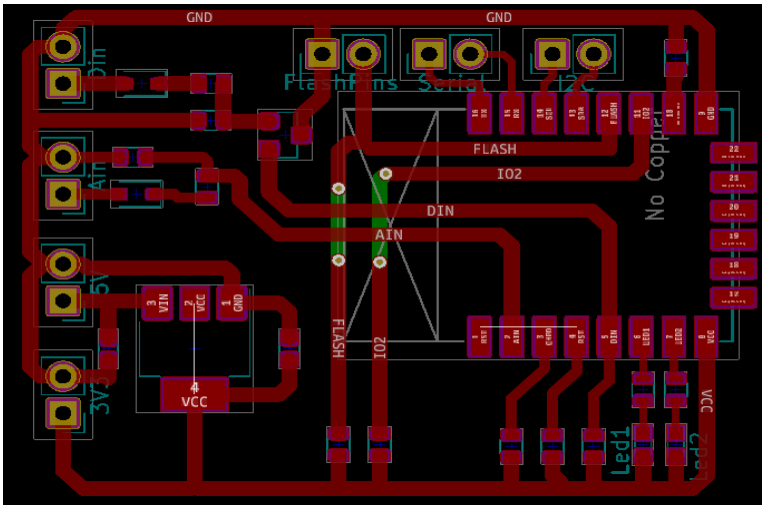

Figura 10: Desenho da placa de circuito impresso no KiCad.

Optou-se por utilizar de componentes com encapsulamento do tipo SMD. Neste tipo encapsulamento os dispositivos são fixos diretamente na superfície da placa de circuito impresso, ao invés dos tradicionais componentes do tipo *through-hole*[2] que necessitam de furos por onde os terminais são passados.

A utilização deste tipo de componentes reduz o tamanho da placa, pois estes componentes geralmente são menores, conforme a FIG. 13**,** com terminais montados junto ao componente e mais curtos. Porém, justamente pelo tamanho reduzido, soldar estes componentes é mais trabalhoso de forma manual, exigindo paciência e um pouco de experiência.

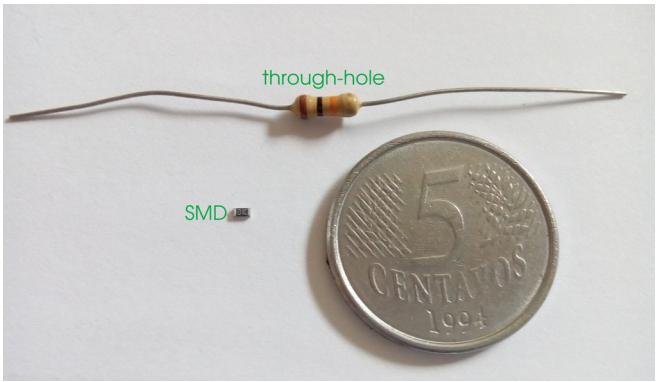

Figura 11: Tipos de componentes e uma moeda de 5 centavos.

2 Através-furo, tradução do autor.

Apesar de o projeto ter sido feito utilizando uma interface 2D, o Kicad permite a visualização da placa em 3D, conforme a FIG. 14. Essa função ajuda na percepção do projeto mais próxima da realidade, permitindo comparar a proporção e distribuição dos componentes na placa.

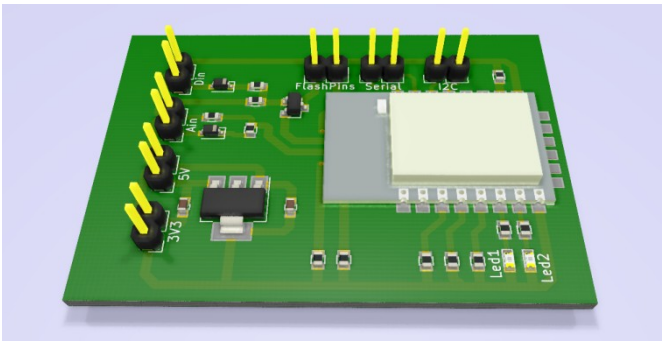

Figura 12: Projeto em 3D da placa dos nós criada através do KiCad.

A confecção da placa de circuito impresso, após o desenvolvimento no software KiCad, foi realizada através do processo de transferência de toner. Este processo inicia-se com a impressão do circuito em uma folha de papel brilhante, como por exemplo, papel fotográfico, como uma impressora a laser.

Posteriormente é necessário transferir o toner que foi depositado pela impressora na impressão para a placa de circuito impresso virgem. Esta parte do processo é realizada através de uma pressa térmica, configurada em 200ºC, que pressiona o papel com o circuito contra a placa, por 200 segundos.

Após esse tempo, são retirados da prensa o papel e a placa já com o circuito transferido. Essa placa é então mergulhada em uma solução de percloreto de ferro, em que será removido o cobre que não está encoberto pela camada de toner. A placa é então finalizada adicionando os componentes, conforme a FIG. 15.

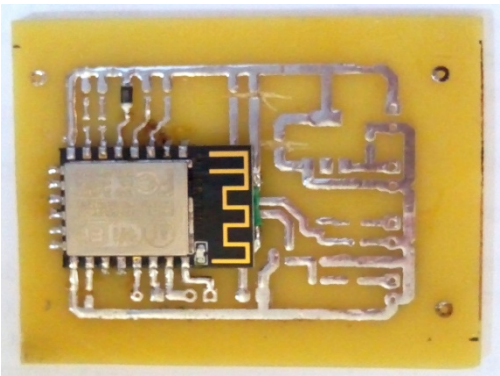

Figura 13: Placa de circuito impresso com já alguns componentes.

Por fim, a placa foi alocada em uma caixa de plástico para dar melhor acabamento e protegê-la contra os fatores que a possam danificar, conforme a FIG. 16. Esta caixa expõe os LEDs de indicação e um conector para a entrada de alimentação e os sinais digital e analógico.

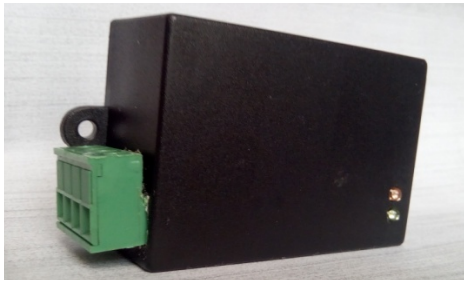

Figura 14: Módulo de nó em uma caixa pronto para utilização.

3.5 Banco de dados da rede

De forma a facilitar a criação e manutenção do banco de dados durante o período de desenvolvimento foi utilizado o *software DB Browser for SQLite.* Este software permite a criação e edição de bancos de dados SQLite, através de uma interface gráfica.

Para criar um novo banco de dados, pressionar o botão "Novo banco de dados" interface do *software*. Após escolher um nome e local para salvar o arquivo do banco de dados, irá aparecer uma tela para criação da primeira tabela do banco de dados, ver FIG. 17.

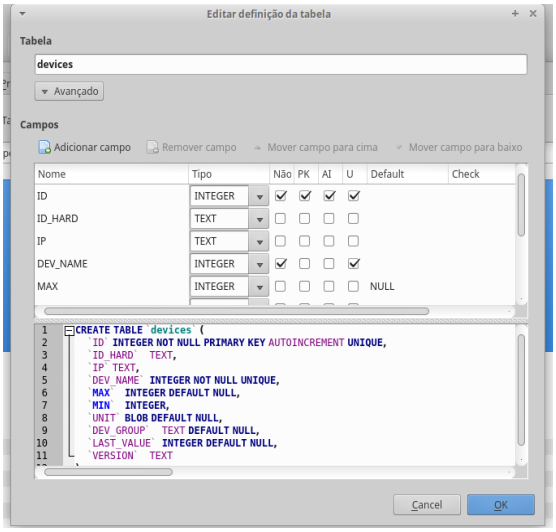

Figura 15: Tela de criação de tabela no DB Browser for SQLite.

Na tela de criação de tabelas, devem-se adicionar os campos, ou seja, as colunas da tabela, que devem ter um nome e um tipo. Além disso, nesta pode-se configurar opções de restrições do campo, isso quer dizer que antes de adicionar uma linha à tabela, estas opções devem ser válidas, ou será retornado um erro.

Além disso, ao adicionar uma linha os valores de cada campo devem ser passados. Caso durante a inserção, um dos dados não for passado, o campo nesta coluna é preenchido com um valor nulo, ou com o valor especificado em “Default”, na tela de criação de tabelas.

O banco de dados criado é composto por duas tabelas: “devices” e “history”. A primeira irá armazenar informações a respeito dos dispositivos conectados à rede, como seu nome e o grupo ao qual este pertence. A segunda será para salvar os dados que estes dispositivos enviam ao servidor.

Entre as colunas presentes na tabela “devices”, há uma para registrar a versão que cada módulo conectado à rede está executando individualmente. Isto permite ao sistema de atualização de *firmware,* garantir que todos os dispositivos estarão na última versão disponível, independentemente se estavam ligados ou conectados à rede no momento em que a atualização ficou disponível. A informação da versão que o módulo está executando chega ao mesmo pacote que os dados do sensor.

Quando o dispositivo realiza uma medida na entrada analógica, o valor será representado em uma escala de 0 a 1024, em que a tensão de 10v é representada pelo valor decimal 931. Para se obter o valor real, na escala da grandeza física medida, deve-se fazer uma conversão, levando em consideração os valores mínimo e máximo na escala real, conforme a equação na Eq.3.

Equação 3: Fórmula para calcular o valor real do dado enviado pelo sensor.

$$valorReal = \frac{valorOriginal \cdot (EscalaFinal_{maximo} - EscalaFinal_{minimo})}{931 + EscalaFinal_{minimo}}$$

Os valores de máximo e mínimo também são armazenados na tabela “devices”, de acordo com o dispositivo cadastrado. Sendo assim, esta tabela apresenta colunas para salvar estes valores, além da unidade da grandeza em que o sensor utilizado estará medindo (ºC no caso de temperatura, Pascal para sensores de pressão, etc.).

Conforme a FIG. 16, a tabela “devices” é constituída por nove colunas e a “history” quatro colunas.

Figura 16: Banco de dados e suas tabelas “devices” e “history”.

A tabela “devices” é composta pelas seguintes colunas:

- **ID:** número de identificação do dispositivo na rede de sensores, selecionado conforme a ordem com que os sensores são adicionados à rede;
- **HARD_ID:** número de identificação do hardware do ESP8266;
- **NAME:** nome dado ao dispositivo na rede, para facilitar a identificação deste por um usuário;
- **UNIT:** unidade específica da grandeza medida pelo sensor, quando é igual a “BOOL” os valores do sensor só podem ser o valor máximo ou o valor mínimo;
- **MAX:** valor máximo que a variável medida pelo sensor poderá atingir, pode ser um texto caso a “UNIT” for “BOOL”;
- **MIN:** valor mínimo que a variável medida pelo sensor poderá atingir, assim como “MAX” pode ser um texto caso “UNIT” for “BOOL”;
- **VERSION:** versão do firmware que o módulo está executando;
- **GROUP:** grupo ao qual o dispositivo pertence, escolhido pelo usuário;
- **LAST_HIST_ID:** informa o número de identificação (“id”) da linha do último dado enviado pelo dispositivo, na tabela de histórico.

Já a tabela de histórico armazena os dados de todos os sensores, em ordem cronológica. A tabela de informações de dispositivos, a “devices”, dispõe de uma coluna para identificar a posição em que se encontra o último dado enviado por cada um dos dispositivos, de forma a aumentar o desempenho da busca.

A tabela “history” é composta pelas colunas:

- **ID:** número de identificação do evento no histórico de dados;

- **HARD_ID:** número de identificação do hardware do ESP8266
- **eventTime:** momento em que ocorreu o evento de dado recebido, salvo na forma de “Unix Time Stamp”;
- **eventValue:** valor enviado pelo dispositivo.

De forma a facilitar o acesso ao banco de dados por outras aplicações, foi desenvolvido um módulo em PHP “QL_DB_Manager constituído por três funções de consulta - “get_all_devices_names”, “get_all_devices_info”e “get_last_value_of” - e três funções de manipulação do banco de dados - “insert_value”, “insert_new_device” e “remove_device”.

A função “get_all_devices_names” faz uma consulta no banco de dados pela coluna “DEV_NAMES” da tabela “devices” no banco de dados. Não recebe nenhum argumento e retorna uma *array*, com os nomes dos dispositivos.

Para se obter todas as configurações dos módulos foi criada a função “get_all_devices_info”. Esta função, também sem argumentos, retorna todas as colunas da tabela “devices” disponível no banco de dados.

Já a função “get_last_value_of” recebe como argumento o nome de um nó de sensor que se deseja consultar o último valor recebido. Ao chamar esta função é verificado se o nome do dispositivo especificado está cadastrado no banco de dados, caso contrário é retornado um valor booleano de falso. Se o dispositivo estiver cadastrado no banco de dados, a função realiza uma consulta na tabela “devices” pelo “LAST_HIST_ID” referente a este. É retornado então o último valor registrado na tabela “history” para este dispositivo.

A função “insert_value” é responsável por guardar um valor no banco de dados para um dispositivo especificado. Recebe como argumentos o valor a ser salvo e o número de identificação do hardware do nó ao qual este pertence. Caso o dispositivo especificado não estiver cadastrado no banco de dados, retorna “falso”, caso contrário, adiciona o dado e retorna “verdadeiro”.

Adicionar um dispositivo na rede de sensores, além de conectá-lo a esta, envolve cadastrá-lo no banco de dados. Para isto pode-se utilizar a função “insert_new_device” que recebe como argumentos as configurações do dispositivo a ser adicionado: nome, número de identificação, máximo, mínimo e unidade da grandeza medida e o grupo ao qual pertence. Retorna “verdadeiro” após salvar no banco de dados ou “falso” em caso de erro, se o nome ou número de identificação já estiverem cadastrados por exemplo.

Para remover um dispositivo já cadastrado foi criada a função “remove_device”, que recebe como argumento o nome do dispositivo. Ao chamar esta função, é verificado se o dispositivo existe, caso contrário já retorna false. Caso o nome do dispositivo especificado esteja cadastrado, é removido da tabela “devices” e “history” todos os registros relacionados.

3.6 Desenvolvimento do Web Service

Os dados recebidos de cada sensor são armazenados em um banco de dados, que se encontra no cartão de memória do *Raspberry Pi*. Para que outros computadores possam acessá-lo foi necessário criar uma interface “Web Service” que permitir não apenas acessar os dados dos sensores, mas também configurar os módulos remotamente.

Esta interface é criada através de um *script* em PHP executado no servidor ao qual tem acesso direto ao banco de dados e, responde às requisições POST ou GET em forma de um JSON.

O funcionamento básico deste *script* é receber as requisições HTTP e executar um tratamento de qual função deve ser executada. Ou seja, a cada vez que um dispositivo envia uma requisição ao *script,* é verificado quais dados estão sendo requisitados e se estão na formatação correta.

Na linguagem de programação *PHP* as requisições *POST* e *GET* recebidas tem os dados salvos em variáveis chamadas “$_POST” e “$_GET”, respectivamente. Esta é do tipo *array* e contém todo o conteúdo recebido. Assim, por exemplo, quando se deseja acessar o conteúdo de uma variável recebida via POST deve-se usar “$_POST[nome_variável]”. Este fato facilita o trabalho com os métodos do protocolo HTTP e acelera o desenvolvimento de projetos.

As mensagens enviadas ao servidor foram divididas em duas categorias: as de consulta no banco de dados, que devem ser recebidas através do método *GET*, e as de inserir dados no mesmo, que devem ser através de *POST*. O algoritmo básico do servidor pode ser observado na FIG. 17.

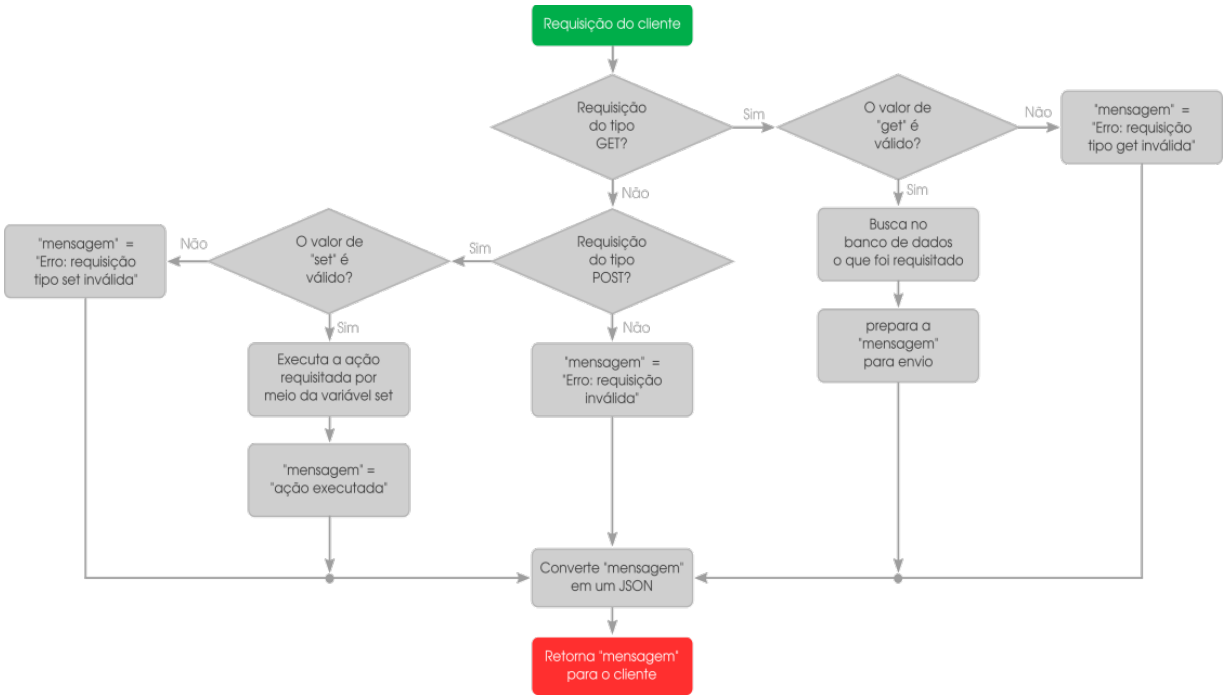

Figura 17: Algoritmo simplificado do sistema REST-service.

Caso a mensagem recebida seja pelo método GET, e a solicitação seja válida, o *script* procura pelos dados solicitados utilizando da classe “QL_DB_Manager”. Após fazer a consulta no banco de dados e obter os dados solicitados, o *script* transforma todos os dados em um JSON para retornar ao cliente.

Se a solicitação for inválida ou os dados requisitados não existirem - por exemplo, a requisição de dados de um sensor em um período de tempo em que não há nenhum a informação salva - o *script* também retorna um JSON, porém com uma identificação de erro.

Para testar o funcionamento do Rest-service para requisições GET, pode-se utilizar um *software* navegador de internet comum. Isso porque as requisições pelo método GET são passadas através da URL adicionando o identificador “?” e as variáveis a serem passadas.

Por exemplo, considerando o *IP* do servidor na rede como “192.168.1.30”, e deseja-se enviar uma variável chamada “get” com um valor igual a “teste” para um *script* PHP “getTeste.php”. A requisição HTTP por meio da GET deve ser feita no navegador através da URL “http://192.168.1.30/getTeste.php?get=teste”.

Podem ser solicitados os seguintes dados, através das mensagens de consulta:

- “last_value”: último dado de um sensor, cujo nome foi especificado através da variável “DEV_NAME”. Na FIG. 18 é mostrado um exemplo de resultado, que

contém o valor do dado recebido - salvo na variável “EVENT_VALUE”, o valor real calculado - salvo na variável “REAL_VALUE”, e o momento em que o dado foi recebido, salvo na variável “EVENT_TIME” no formato Unix-Time;


```
{ ▼ 2 properties, 124 bytes
  "required": "last_value",
  "result": { ▼ 4 properties, 89 bytes
    "EVENT_VALUE": 0,
    "EVENT_TIME": 1497216593,
    "DEV_NAME": "INDUTIVO",
    "REAL_VALUE": "Desativado"
  }
}
```


Figura 18: Exemplo de JSON do último valor de um sensor.

- “all_devices_info”: informações de configurações do sensor. Resultado com o nome, máximo, mínimo e unidade da variável medida, número de identificação, grupo de sensores ao qual o nó pertence e versão do firmware. Na FIG. 19 é mostrado um exemplo de resultado;


```
{ ▼ 2 properties, 259 bytes
  "required": "all_devices_info",
  "result": [ ▼ 2 items, 218 bytes
    { ▼ 7 properties, 113 bytes
      "ID": 1,
      "DEV_NAME": "INDUTIVO",
      "MIN": "Desativado",
      "MAX": "Ativo",
      "UNIT": "BOOL",
      "DEV_GROUP": "SALA2",
      "VERSION": "1.2"
    },
    { ▼ 7 properties, 102 bytes
      "ID": 2,
      "DEV_NAME": "DISTANCIA",
      "MIN": "0",
      "MAX": "1000",
      "UNIT": "cm",
      "DEV_GROUP": "SALA2",
      "VERSION": "1.2"
    }
  ]
}
```


Figura 19: Exemplo de JSON com os dados de todos os dispositivos.

- “all_devices”: Lista dos nomes de sensores registrados no banco de dados. Na FIG. 20 é mostrado um exemplo de resultado;

```
{ ▼ 2 properties, 91 bytes
  "required": "all_devices",
  "result": [ ▼ 5 items, 55 bytes
    "DISTANCIA",
    "INDUTIVO",
    "LUMINOSIDADE",
    "TEMP1",
    "TEMP2"
  ]
}
```


Figura 20: Exemplo de JSON com os nomes de todos os dispositivos.

Caso a mensagem pelo método *POST*, o *script* também verifica se é uma solicitação válida e prepara os dados para serem salvos no banco de dados.

Para testar as requisições por meio do método POST, deve-se utilizar um formulário HTML. Este será responsável por enviar as variáveis determinadas ao servidor para análise.

Por exemplo, na FIG. 21 é apresentado um código em HTML que cria um formulário. Na linha um, define-se a URL de destino da mensagem, no caso “http://192.168.1.30/postTeste.php”, e o método HTPP de envio, no caso POST.

```
1 <form action="http://192.168.1.30/postTeste.php" method="post">
2     <label>Set:</label>
3     <input id="set" type="text" />
4     <button type="submit">Enviar mensagem</button>
5 </form>
```

Figura 21: Exemplo de código HTML para criação de um formulário.

As variáveis a serem enviadas por meio deste formulário têm seus valores alterados através do campo de entrada de texto, que aparecem ao acessar este formulário por meio de um navegador de internet conforme a FIG. 22. Para adicionar mais variáveis na requisição é necessário apenas adicionar novos campos definidos como *“input”,* cujo *“id”* será o nome da variável.

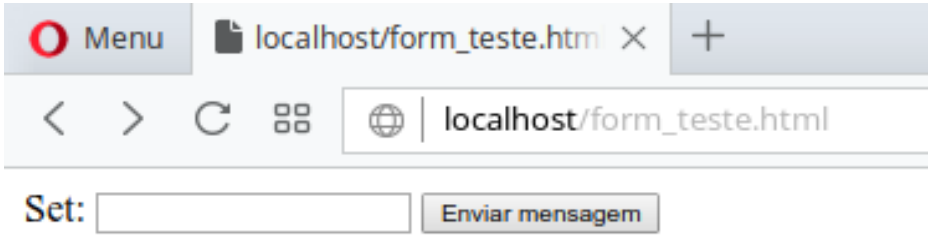

Figura 22: Formulário de teste, ao acessar pelo navegador de internet.

Podem ser realizadas as seguintes requisições de salvar dados no banco de dados:

- “new_value”: salva um valor, especificado através da variável “value”, para um dispositivo, cujo número de identificação foi especificado através de “hardid”. Conforme a FIG. 23, após salvar o dado, será retornado um JSON com o id do dispositivo, o valor salvo, e se a operação foi concluída com sucesso, através da variável “result”;


```
{ ▼ 4 properties, 72 bytes
  "required": "new_value",
  "ID_HARD": "8402791",
  "value": "123",
  "result": true
}
```


Figura 23: JSON retornado ao solicitar a inserção de um novo valor.

- “new_device”: adiciona um novo dispositivo ao banco de dados. Devem ser especificados o nome - através da variável “devName”, o ID do hardware - através da variável “hardid”, o grupo do sensor - com a variável “devGroup”, o valor mínimo e o máximo - através de “min” e “max” respectivamente, e a unidade de medida - através de “unit”. Retorna os valores que foram adicionados ao banco de dados na variável “input” e o resultado que pode ser “verdadeiro”, caso adicionou o sensor com sucesso ou “falso” caso contrário, conforme FIG.24.


```
{ ▼ 3 properties, 145 bytes
  "required": "new_device",
  "input": { ▼ 6 properties, 97 bytes
    "devName": "Distancia",
    "hardid": "112433",
    "devGroup": "Teste",
    "min": "0",
    "max": "10",
    "unit": "Metros"
  },
  "result": true
}
```


Figura 24: JSON Retornado ao solicitar a inserção de um novo dispositivo.

3.7 Interface de gerenciamento

Com o objetivo de melhorar o gerenciamento da rede de sensores, além de ter uma visão administrativa da mesma, foi criada uma interface de gerenciamento, chamada de “Painel QuickLink”, ver FIG. 25. A interface de controle disponibiliza a visualização dos últimos valores de cada sensor configurado na rede, além da possibilidade de configurar novos.

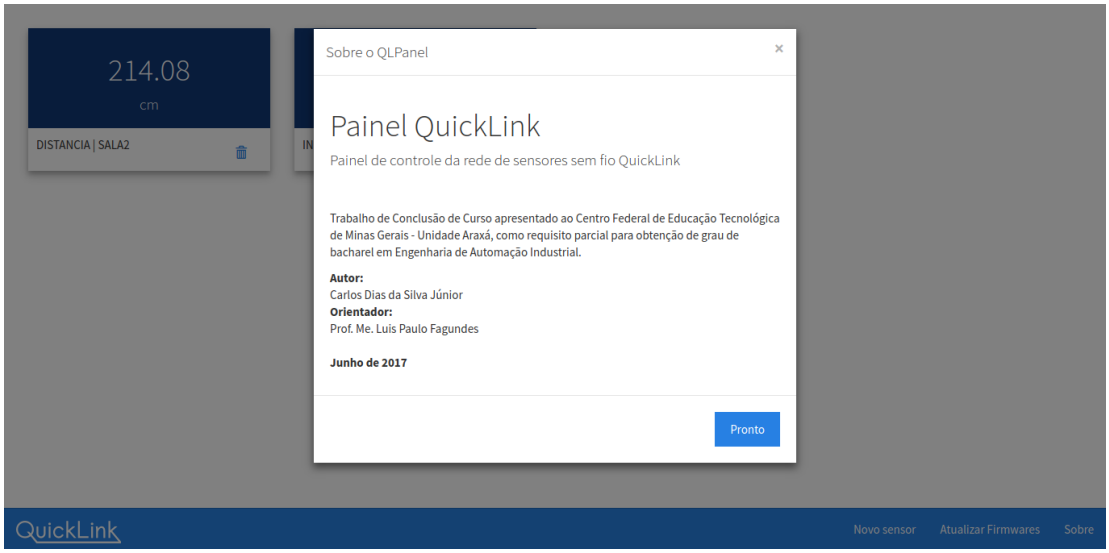

Figura 25: Janela flutuante “Sobre o QLPanel” na interface de gerenciamento.

Esta interface foi completamente criada com HTML, CSS e JavaScript, o que a torna independente da plataforma em que está sendo executada, pois depende apenas de um navegador de internet. Com o objetivo de agilizar o desenvolvimento do projeto foi utilizado a biblioteca JQuery e a *framework* Bootstrap.

Ao carregar a página um *script* em JavaScript envia uma mensagem via GET ao arquivo “restservice.php” no servidor requerendo os dados de todos os sensores já cadastrados, através do valor “all_devices_info” - conforme item 3.6 deste trabalho.

Com os dados de todos os sensores cadastrados no banco de dados, o mesmo *script* em JavaScript, adiciona na página as informações básicas dos sensores: nome, grupo e valor atual.

Todas as mensagens dos nós dos sensores são enviadas a um script PHP responsável por salvá-los no banco de dados, conforme 3.6. Quando um

sensor que não está cadastrado no banco de dados está conectado à rede, as mensagens enviadas ao servidor são salvas apenas em um arquivo de registro.

Este arquivo possibilita verificar as mensagens que estão chegando ao servidor, sem guardá-las no banco de dados. Além disso, foi pode-se verificar cada um dos números de identificação únicos de cada um dos chips, pois esta informação está no corpo das mensagens.

Ao clicar em “Adicionar sensor” o formulário envia uma mensagem via POST ao REST Service com um campo chamado “set”, não visível ao usuário, com valor igual a “new_device”. Este campo permite ao *REST service* identificar que se deseja adicionar um novo dispositivo ao banco de dados da mensagem.

Ao adicionar o novo sensor, a página é recarregada com o novo sensor cadastrado já visível, mostrando o último valor lido, atualizado em tempo real, seu nome e grupo ao qual pertence, e um botão com um ícone de lixeira para remover o sensor, conforme FIG. 26.

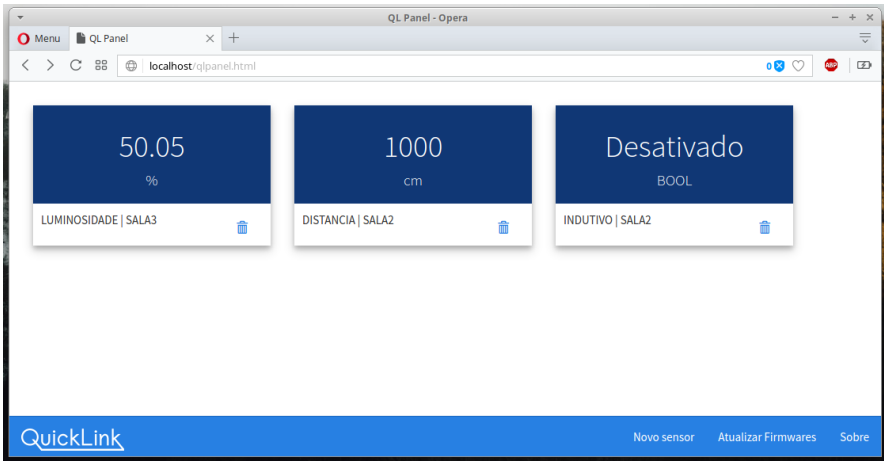

Figura 26: Interface de gerenciamento da rede de sensores.

Os dados de cada um dos sensores são atualizados a cada 250ms através de uma função temporizada em JavaScript. Esta função faz requisições ao servidor dos dados de todos os sensores e atualiza o campo do valor atual.

Para adicionar um novo sensor à rede foi desenvolvida uma janela flutuante de configuração, disponível através do botão “Novo sensor” na barra de menus inferior na interface.

Esta janela é composta por um formulário HTML, com entradas para os dados de configuração do novo sensor, conforme a FIG. 27. Todos os campos deste devem ser obrigatoriamente preenchidos para que seja possível enviar o formulário.

A validação foi feita utilizando apenas o atributo “required” na *tag* de cada campo do formulário.

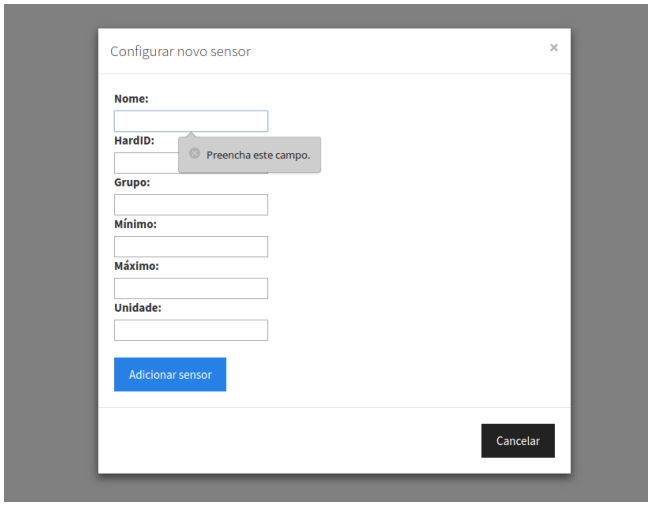

Figura 27: Janela flutuante para configuração de novo sensor.

4 RESULTADOS E DISCUSSÕES

4.1 Custos do projeto

Para se desenvolver o protótipo apresentado neste trabalho, foi necessário adquirir diferentes componentes, em lojas tanto no Brasil quanto importado. O preço pago pelo NodeMCU em agosto de 2016 foi de R$ 37,50. O Raspberry Pi 2 foi adquirido em maio de 2015 pelo preço de R$244,99.

As placas dos nós são compostas por vários componentes. O ESP12E foi adquirido em loja no Brasil por R$12,00. Os outros componentes - resistores de 10KΩ, resistores de 100KΩ, MMBT2222, LM117 e capacitores de 10 µF - foram adiquiridos em lotes de 200 por meio de importação. Devido a isto, o custo de todos esses componentes somados é de menos de R$2,00.

4.2 Consumo de energia

Dependendo da aplicação, pode ser necessário utilizar o ESP8266 conectado a uma bateria, como no projeto de Mainwaring et al. (2002). Por isso, a gestão de energia é uma parte muito importante no desenvolvimento das redes de sensores sem fio.

O ESP8266 tem três modos de economia de energia: “modem-sleep” - em que apenas o WiFi é desligado -, “light-speed”- em que o processador interno entra em modo suspenso - e “deep-sleep” - que pode ser chamado apenas pelo usuário e desliga todo o módulo ESP8266.

Devido a esse motivo, foi mensurado o consumo de corrente, adicionando um multímetro no modo de corrente entre o positivo da fonte e o positivo da entrada do regulador de tensão da placa dos nós.

Foram feitos testes no modo de “deep-sleep” e o modo de execução normal do firmware, e observou-se que a corrente varia durante todo o período de execução. Os valores de corrente minima e maxima medidos tiveram seus valores anotados na TAB. 2.

Tabela 2: Corrente elétrica em dois dos modos de operação do ESP8266

Modo	Corrente mínima	Corrente máxima
Executando Firmware conectado a rede sem fio	68 mA	77 mA
deep-sleep	17 µA	17 µA

Os testes foram realizados utilizando o mesmo *firmware* que envia os dados para o servidor. A corrente máxima foi atingida nos momentos em que o ESP8266 estava enviando pacotes. Já o modo “deep-sleep” foi testado com um *firmware* que conectava à rede sem fio e depois de cinco segundos, entrava no modo de economia de energia.

É possível comutar o módulo entre os modos por tempos pré-determinados permitindo que sejam enviados os dados dos sensores e depois desligar o ESP8266 para economizar energia. Para isso, deve-se conectar o pino GPIO16 ao pino de *reset.*

É importante salientar que para escolher os tempos de comutação é necessário levar em consideração a natureza da aplicação, pois os períodos de economia de energia devem ser ajustados de acordo com a frequência que se espera do sensor.

Aplicações em que as grandezas das variáveis medidas não têm grandes variações no tempo, como temperatura ou umidade do ar, podem ser beneficiadas pelo uso do modo de economia de energia.

Por exemplo, um sensor de temperatura pode ser ajustado para transmitir os valores medidos em intervalos de vinte segundos por ter uma baixa variação no tempo da grandeza medida.

Além disso, caso os sensores forem em ambientes externos e expostos à luz solar, é possível utilizar de pequenos painéis solares. Para isto também se deve levar em consideração o consumo de energia dos sensores envolvidos, para a escolha da bateria ou de “supercapacitores” de carga e painel solar.

4.3 Utilização com sensores industriais

Ao utilizar um sensor analógico, todos os valores do conversor AD terão uma margem de erro, devido à precisão do conversor analógico digital, além do divisor de tensão. Esse erro aumenta proporcionalmente ao valor de entrada como é possível visualizar pelo gráfico da FIG. 28.

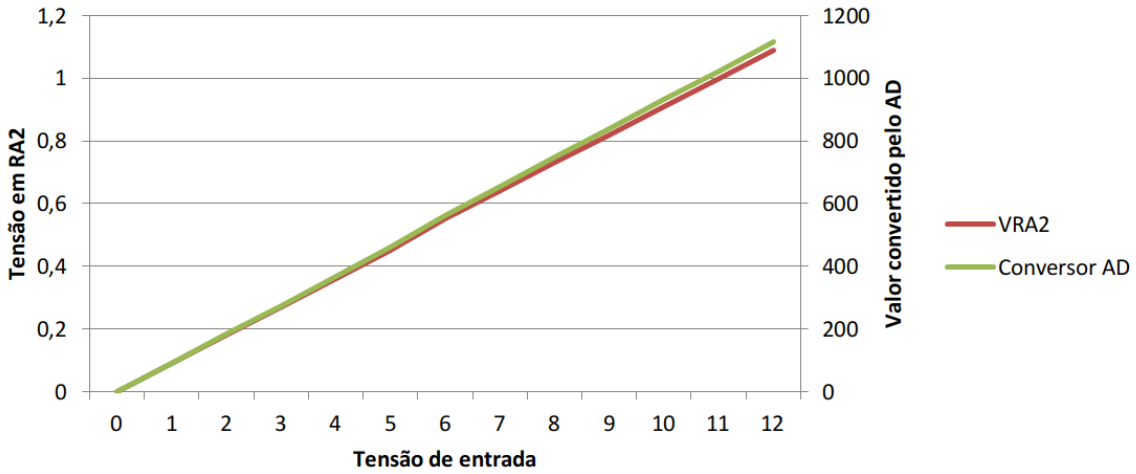

Figura 28: Gráfico da relação entre a tensão de entrada a tensão em RA2.

A precisão do conversor analógico-digital do ESP8266 é de 10 bits o que resulta em uma precisão mínima de aproximadamente 977µV. Porém uma variação de 1mV na entrada, conforme Eq. 4, resultará em uma variação de 90,9 µV no resistor RA2, sendo impossível de ser medida pelo conversor.

Equação 4: cálculo da variação de 1mV na entrada em relação à tensão em RA2

$$I_{total} = \frac{V_{entrada}}{R_{A1}} + R_{A2} \quad \Rightarrow \quad I_{total} = \frac{1\,mV}{100\,k + 10\,k} \quad \Rightarrow \quad I_{total} = 9{,}09\,nA$$

$$V_{RA2} = R_{A2} \cdot I_{total} \quad \Rightarrow \quad V_{RA2} = 10\,k \cdot 9{,}09\,n \quad \Rightarrow \quad V_{RA2} = 90{,}9\,\mu V$$

Sendo assim, a menor variação possível de ser medida pelo conversor analógico-digital pode ser calculada utilizando a precisão mínima como sendo o valor de 11mV.

Considerando os fatores apresentados, pode ser necessária a utilização de um módulo externo de conversor analógico-digital, caso a aplicação necessite de maior precisão nos valores medidos. Estes dispositivos externos podem ser conectados ao módulo do nó por meio da porta I2C disponível na placa do mesmo.

5 CONCLUSÕES

Algumas características do protótipo apresentado devem ser reformuladas de acordo com a aplicação da RSSF. Isto é devido à dificuldade de se criar um sistema que consiga atender a diferentes demandas.

Conforme o item 4.1 deste trabalho, os custos por nó deste trabalho foram baixos quando comparados aos dos autores citados. Quanto ao tamanho das placas dos nós, este ainda pode ser reduzido utilizando de placas de dupla camada o que, em contrapartida, aumenta os custos de produção, e ao mesmo tempo dificulta a confecção por métodos manuais, como o apresentado neste trabalho.

A respeito da escalabilidade, um fator que influencia muito é o alcance do roteador Wireless, que deve cobrir toda a área da aplicação da RSSF. Uma forma de aumentar o alcance é utilizar mais pontos de acesso à rede, espalhados em uma grande área.

Também é possível utilizar o Raspberry Pi como servidor temporário, para um conjunto de sensores. Neste caso, os dados são armazenados por certo período para serem enviados posteriormente ao servidor principal. Esta abordagem permite que a carga de trabalho de trabalho do servidor seja reduzida e possibilita a criação de uma rede com grandes proporções.

A possibilidade de integração com sistemas externos foi atingida com a criação do sistema de *REST Service*: através deste os dados dos sensores estão disponíveis para consulta, além de também ser possível configurá-los.

O desenvolvimento da interface de gerenciamento criada para este trabalho demonstrou que é possível utilizar os dados dos sensores, independentemente da plataforma em que se está desenvolvendo. Algumas funções podem ser desenvolvidas nesta interface tais como gráficos em tempo real para as variáveis analógicas e controle de acesso por meio de *login*.

Além disso, é possível configurar os nós para que enviem os dados para uma plataforma como a Xively, descrita no item 2.2 deste trabalho. Para isto é necessário que a rede em que os sensores estão conectados tenha acesso à internet, além de algumas mudanças no firmware para que os nós enviem os dados para esta plataforma.

Os resultados de consumo de energia demonstram que os nós podem ser limitados a aplicações que tenham disponibilidade de uma fonte de energia

constante se não for realizado um controle de eficiência energética, conforme discutido no item 4.2.

Outro assunto para trabalhos futuros, refere-se à segurança dos dados: este trabalho limitou-se a utilizar como único sistema de segurança a chave de acesso à rede sem fio. Qualquer dispositivo conectado à rede tem acesso às informações dos sensores, criando uma grande brecha para utilizadores não autorizados.

Pode-se estudar a possibilidade de encriptação ponta a ponta dos dados entre o servidor e cada nó da rede. Este recurso irá consumir mais recursos dos processadores de cada nó, e consequentemente mais energia, porém permitiria mais segurança para, por exemplo, um sensor controlar um atuador na rede.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDMEZIEM, Riad; TANDJAOUI, Djamel. Internet of Things: Concept, Building blocks, Applications and Challenges. **arXiv preprint arXiv:1401.6877**, 2014.

AKYILDIZ, Ian F. et al. Wireless sensor networks: a survey. **Computer networks**, v. 38, n. 4, p. 393-422, 2002.

BRÖRING, Arne et al. New generation sensor web enablement. **Sensors**, v. 11, n. 3, p. 2652-2699, 2011.

CHONG, Chee-Yee; KUMAR, Srikanta P. Sensor networks: evolution, opportunities, and challenges. **Proceedings of the IEEE**, v. 91, n. 8, p. 1247-1256, 2003.

FERREIRA, Ivo António Pereira Faustino et al. Sistemas de controlo e supervisão de sistemas embebidos: tipo SCADA. 2012.

GOMES, Ruan D. et al. Desafios de redes de sensores sem fio industriais. **Revista de Tecnologia da Informação e Comunicação**, v. 4, n. 1, 2014.

GUNGOR, Vehbi C.; HANCKE, Gerhard P. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. **IEEE Transactions on industrial electronics**, v. 56, n. 10, p. 4258-4265, 2009.

JANGAM, Keerthi Bhushan et al. Wireless Sensor Network Design for Tactical Military Applications. **International Journal of Applied Sciences, Engineering and Management**, v. 05, n. 2, p. 05 - 08, 2016.

KO, Jeong Gil et al. Wireless sensor networks for healthcare. **Proceedings of the IEEE**, v. 98, n. 11, 2010.

KORTUEM, Gerd et al. Smart objects as building blocks for the internet of things. **IEEE Internet Computing**, v. 14, n. 1, p. 44-51, 2010.

LOUREIRO, Antonio A. F. et al. Redes de sensores sem fio. In: **Simpósio Brasileiro de Redes de Computadores (SBRC)**. Local: s.n., 2003.

MAINWARING, Alan et al. Wireless sensor networks for habitat monitoring. In: **Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications**. ACM, 2002.

MIORANDI, Daniele et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, v. 10, n. 7, p. 1497-1516, 2012.

RUIZ, Linnyer Beatrys et al. Arquitetura para Redes de Sensores Sem Fio. In: **22º Simpósio Brasileiro de Redes de Computadores (SBRC),** 2004.

SILVA, Marcel S.; FRUETT, Fabiano. Rede de sensores sem fio de baixo custo para monitoramento ambiental. In: **Anais: XVIII Congresso Brasileiro de Automática. Bonito**, 2010.

WAGH, Kishor; THOOL, Ravindra. A comparative study of soap vs rest web services provisioning techniques for mobile host. **Journal of Information Engineering and Applications**, v. 2, n. 5, p. 12-16, 2012.