**Aplicação Web para Monitoramento de Sensores utilizando PIC 16F877A e Node.js**

Gabriela Gomes dos Santos, Vitor Bruno de O. Barth

IFMT – DAI

gabi.gsantos@hotmail.com, vbob@vbob.com.br

***Resumo –*** *A Internet das Coisas está cada vez mais presente no dia-a-dia da sociedade. Uma parte essencial desta é o monitoramento de sensores analógicos. Erroneamente, muitos trabalhos utilizam placas de prototipagem completas como nós de redes de sensores. Este trabalho mostra que utilizar um microcontrolador individual para obtenção de dados de sensores e transmissão destes dados sem fio até um ponto de entrada, reduzindo a complexidade e custos de formação de uma rede de sensores. Como resultado, foi obtido um sistema estável e robusto, capaz de realizar a transmissão sem fio de dados de sensores conectados a um PIC 16F877A, e acompanhá-los em tempo online via web.*

1. INTRODUÇÃO

Estamos vivenciando o nascer da nova era da Internet das Coisas(IoT, do inglês *Internet of Things*). IoT se refere à interconexão dos dispositivos do dia-a-dia, que podem colher informações ou atuar sobre o ambiente [1]. Uma de suas partes principais são as Redes de Sensores sem Fio, que estão sendo aplicada para monitoramento de indústrias, meio ambiente, saúde, etc. [2].

O *hardware* que compõe Redes de Sensores sem Fio é extremamente simples: microcontroladores de baixo consumo de energia, acoplados a interfaces analógicas para leitura de sensores e um transmissor sem fio formam os *nós* desta rede. Os dados transmitidos por cada um destes nós é recebido em um *gateway* (do inglês, Porta de Entrada), que processa estes dados de acordo com a aplicação planejada [3].

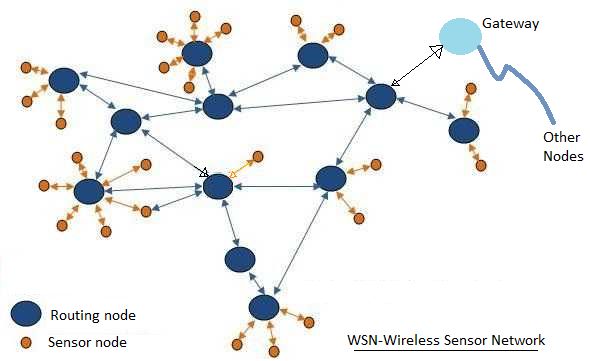


Figura 1 - Rede de Sensores. Fonte: RF Wireless World.

Ainda que comumente utilizados como *nós* da Rede de Sensores Sem Fio, placas de prototipagem como *Raspberry Pi* e *Arduino* [3]–[6]são pouco apropriados para este fim, pois estas agregam complexidade desnecessária ao sistema e consequentemente possuem dimensões e consumo de energia maiores que sistemas mais simples. Placas mais simples, projetadas para o fim específico ao qual se destinam são mais eficientes e mais fáceis de serem integradas ao ambiente [3].

Para garantir o funcionamento adequado da Rede de Sensores sem Fio, além do *hardware* descrito, é necessário uma solução em *software* para o *gateway* capaz de lidar com grande fluxo de dados e que seja de fácil integração com novos nós [6]. Uma ferramenta flexível e de alta performance que têm se destacado no mercado é o servidor *web* Node.js [7], [8].

Os avanços na linguagem JavaScript propiciaram o surgimento de novas formas de interação entre a web e o mundo real. JavaScript, é uma linguagem de programação interpretada, criada em 1995 e amplamente usada no lado do cliente, com a intenção de executar, dentro do navegador, funções que alterem um website sem a necessidade de comunicação com o servidor. A fim de melhorar a velocidade de execução de códigos JavaScript, a Google lançou, em 2008, o motor de compilação Chrome V8, que compila diretamente o código em JavaScript para linguagem de máquina, o qual propiciou o surgimento de diversos projetos de software, como o servidor web Node.js [8].

O principal benefício do Node.js é se manter estável frente a aplicações de tempo real com tráfego intenso de dados, que é obtido utilizando programação Assíncrona Orientada a Eventos, e, portanto, consegue responder a mais requisições em menor tempo em comparação com outros servidores web [9].

Com base no exposto, este trabalho propõe um sistema de monitoramento de nós de uma Rede de Sensores sem Fio utilizando o servidor *web* Node.js.

O restante do trabalho é divido como segue: a seção 2 descreve os equipamentos utilizados, enquanto a seção 3 o protocolo de transmissão utilizado*.* A seção 4 descreve a solução em *software*, e na seção 5 são apresentados as dificuldades e o resultado do trabalho.

1. EQUIPAMENTOS

Todo o sistema foi planejado e desenvolvido de modo a ser possível monitorar a leitura de dois sensores, posicionados a 1 metro de distância do *gateway*, utilizando um conjunto transmissor-receptor de rádio a 433MHz*.*

O ponto de recepção no *gateway* utiliza um conversor USB-TTL para leitura os dados seriais. Tanto o transmissor quanto o receptor utilizam um microcontrolador PIC 16F877A para comunicação. Entretanto, o microcontrolador utilizado para a recepção não é obrigatório, visto que é possível conectar o receptor RF diretamente ao conversor USB-TTL.

Para simular sensores reais, foram utilizados dois potenciômetros lineares de 10 KΩ, mas estes podem ser substituídos por qualquer sensor analógico.

O custo aproximado e a lista de componentes utilizada são apresentados no Quadro 01. Não foi avaliado o custo de montagem do circuito e itens de suporte, como placas, matrizes de contato, solda, gravadora para o microcontrolador, fios, etc.

Quadro 1 - Material Utilizado

|  |  |  |  |
| --- | --- | --- | --- |
| **Equipamento** | **Qtd.** | **Valor Un. (R$)** | **Total**  **(R$)** |
| PIC 16F877A | 2 | 15,00 | 30,00 |
| RF 433MHz | 1 | 8,00 | 8,00 |
| USB-TTL | 1 | 8,50 | 8,50 |
| Potenciômetro | 2 | 2,50 | 5,00 |
| **Total (R$)** | | | **51,50** |

1. TRANSMISSÃO

O sistema proposto foi construído para transmitir e receber dados de acordo com o seguinte protocolo:

Valor Lido + Indicador de Origem + Indicador de Fim

Por estar se utilizando uma baixa taxa de dados – 1200 baud/s a 9 *bits*/baud, fornecendo uma taxa máxima de 10,8 kbps – em uma banda de 433MHz, e o receptor conectado a um *gateway* de alta performance, preferiu-se regular a transmissão em forma de fluxo de *string*. De modo a diminuir perda de pacotes e dados inconsistentes, todos os quadros possuem um indicador de paridade.

Sendo assim, para uma leitura analógica com precisão de 10 bits, são necessários pelo menos 6 quadros na forma “8 *bits* + 1 *bit* de paridade”– por exemplo, o fluxo “1023A\n” indicaria a leitura do valor 1023 na origem A, que por ser uma cadeia de caracteres, cada um dos dígitos formaria um quadro de 9 *bits.*

Foi utilizada uma taxa de leitura analógica de 500Hz. Como há somente um conversor Analógico Digital presente no microcontrolador, a taxa de leitura é dividida entre todos os sensores, sendo então a taxa individual de amostragem 500Hz/Numero de Sensores.

Com esta configuração de rede, a banda B teórica necessária pode ser calculada através de:

Entretanto, esta banda é puramente teórica: na prática, a taxa de dados utilizada é de apenas 100 bps, devido a perdas na transmissão e recepção, limitando a frequência de amostragem para 12 Hz aproximadamente.

1. SOFTWARE

O *software* utilizado em ambos os microcontroladores é apenas para redirecionamento: no transmissor, é feita a leitura sequencial circular dos sensores, e os resultados são convertidos para um fluxo de *strings* que atende ao protocolo de transmissão. Todo o trabalho de decodificação e mapeamento é realizado no *gateway*. Além disso, como há um protocolo bem definido, pode ser feita uma avaliação dos dados recebidos, permitindo detectar desconexão ou ruído no canal.

A aplicação desenvolvida é baseada na classe *LinearSensor*, que possuí os seguintes atributos:

* Nome
* Indicador de Origem
* Valor de Leitura Mínimo
* Valor de Referência Mínimo
* Valor de Leitura Máximo
* Valor de Referência Máximo
* Tamanho do *Buffer* de Amostras

Com isto, é possível mapear o valor recebido ao sensor ao qual se refere, e estabelecer a relação de proporcionalidade entre o valor lido e a grandeza especificada.

Como saída, optou-se por exibir os valores lidos em um gráfico na *web*, entretanto este pode ser facilmente substituído por um Banco de Dados.

Para exibir os gráficos na *web*, construiu-se uma API genérica que transmite os *buffers* de amostras de todos os sensores na forma de um vetor, e pode ser visto na Figura 2.



Figura 2 - Resposta da API. Elaborado pelos autores.

A partir disso, utilizando o *framework* de construção de gráficos C3.js, pode-se plotar os gráficos com estes dados na *web,* sendo um exemplo mostrado na Figura 3.

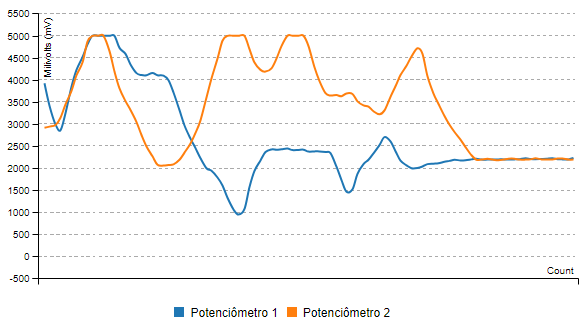


Figura 3 - Gráficos gerados na Web. Elaborado pelos autores.

Para atualização deste em tempo *online*, utilizou-se a biblioteca AJAX, que realiza chamadas no plano de fundo a cada intervalo definido. Para garantir fluidez à aplicação e diminuir a necessidade de banda, optou-se por uma taxa de atualização de 200ms, entretanto esta pode ser alterada de acordo com a necessidade.

1. CONSIDERAÇÕES FINAIS

Com este trabalho foi possível unificar tudo o que foi visto durante o curso de Engenharia da Computação até o presente semestre: conhecimentos em programação em alto e baixo nível, comunicação de dados e prototipagem de circuitos foram necessárias para que se pudesse elaborar o trabalho proposto.

Garantir a estabilidade e consistência deste sistema foi um tanto quanto trabalhoso, assim como estabelecer um protocolo de comunicação simples, mas que se adaptasse ao *hardware.*

Como resultado, foi obtido um sistema robusto e flexível, capaz de atender os requisitos do projeto. Entretanto as baixas taxas de transmissão não o fazem apropriado para todas as aplicações.

1. REFERÊNCIAS

[1] O. León, J. Hernández-Serrano, and M. Soriano, “Securing cognitive radio networks,” *Int. J. Commun. Syst.*, vol. 23, no. 5, pp. 633–652, 2010.

[2] J. Tang, A. Liu, M. Zhao, and T. Wang, “An Aggregate Signature Based Trust Routing for Data Gathering in Sensor Networks,” *Secur. Commun. Networks*, vol. 2018, no. 1, 2018.

[3] S. Ferdoush and X. Li, “Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications,” *Procedia Comput. Sci.*, vol. 34, pp. 103–110, 2014.

[4] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C.-H. Lung, “Smart Home: Integrating Internet of Things with Web Services and Cloud Computing,” *2013 IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, no. November, pp. 317–320, 2013.

[5] J. Zhou *et al.*, “CloudThings: A common architecture for integrating the Internet of Things with Cloud Computing,” *Proc. 2013 IEEE 17th Int. Conf. Comput. Support. Coop. Work Des. CSCWD 2013*, pp. 651–657, 2013.

[6] V. Vujović and M. Maksimović, “Raspberry Pi as a Sensor Web node for home automation,” *Comput. Electr. Eng.*, vol. 44, pp. 153–171, 2015.

[7] J. Bermúdez-Ortega, E. Besada-Portas, J. A. López-Orozco, J. A. Bonache-Seco, and J. M. D. La Cruz, “Remote Web-based Control Laboratory for Mobile Devices based on EJsS, Raspberry Pi and Node.js,” *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 158–163, 2015.

[8] P. Ratanaworabhan, B. Livshits, and B. G. Zorn, “JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications,” *WebApps*, pp. 3–3, 2010.

[9] R. R. McCune, “Node.js Paradigms and Benchmarks,” *Striegel, Gr. Os F*, vol. 11, pp. 1–6, 2011.

ANEXO I - SOFTWARE DO MICROCONTROLADOR

Disponível em: <https://github.com/vbob/PicPlotJS>