Capítulo 6

Introdução Prática à Internet das Coisas: Prática utilizando Arduíno e Node.js

Cintia Carvalho Oliveira, Daniele Carvalho Oliveira, João Carlos Gonçalves, Júlio Toshio Kuniwake

Abstract

The Internet of Things brings the idea of "all connected", from television sets that choose the schedule according to the viewer's preferences to cars that select the faster path to the desired point by its driver. With the expansion of Web access not only computing devices such as laptops, personal computers and smartphones but also equipment such refrigerators, television, among others, the possibilities of communication and automation expand and cities, homes and smart environments became one reality. This work aims to explore the potential of this technology by presenting the necessary steps to develop a simple application that uses the Arduino platform and framework Node.js.

Resumo

A Internet das Coisas traz a ideia de "tudo conectado", desde aparelhos de televisão que escolhem a programação de acordo com a preferência do espectador a carros que selecionam o trajeto mais rápido até o ponto desejado por seu condutor. Com a expansão do acesso à Web não só de dispositivos computacionais como notebooks, computadores pessoais e smartphones mas também de equipamentos como geladeiras, televisão, entre outros, as possibilidades de comunicação e automatização se ampliam e cidades, casas e ambientes inteligentes passam a ser uma realidade. Este trabalho tem como objetivo explorar as potencialidades desta tecnologia apresentando os passos necessários para o desenvolvimento de uma aplicação simples que utiliza a plataforma Arduino e o framework Node.js.

6.1. Introdução

Internet das Coisas (IoT – *Internet of Things*) é uma estrutura de rede global e dinâmica que possibilita a autoconfiguração, baseando-se em protocolos padronizados de

comunicação, onde seus componentes físicos e virtuais possuem identidades, assim usando interfaces inteligentes ligadas a Internet [Atzori et al. 2010].

Na Internet das Coisas seus componentes são capazes de interagir entre si, deste modo trocando informações em diferentes ambientes além de reagir automaticamente aos eventos do mundo real, deste modo não havendo exatamente uma intervenção direta do ser humano em sua infraestrutura. A IoT é uma tecnologia que tem como objetivo conectar diferentes softwares, dispositivos eletrônicos, máquinas industriais, "coisas" no termo geral, tudo fazendo uso de métodos dinâmicos, além do uso de sensores, nanotecnologia, *wireless*, componentes conectados via Internet.

Internet of Things exige uma forma de segurança feita sob medida, pois ao contrário de computadores tradicionais, seus elementos são constituídos de pouca capacidade de energia, processamento e memória [Atzori et al. 2010], assim realizando seu trabalho de forma colaborativa. Podemos visualizar a IoT como uma tecnologia "maciça", uma tecnologia que engloba diferentes conceitos para se tornar único, a principalmente três componentes que são combinados para poder haver uma aplicação IoT, que são:

Dispositivos: Os dispositivos são itens que podem ter diferentes tamanhos, grandes a minúsculos, porém não é o tamanho do dispositivo que faz a diferença e sim como esses dispositivos são equipados para favorecer a comunicação entre seus diferentes componentes, tais dispositivos podem ser equipados com sensores, chips, antenas, dentre outros.

Redes de Comunicação: A comunicação via Internet que conhecemos pode ser usada para a IoT, no entanto as redes atuais possuem alcance limitado, obrigando que certas aplicações façam uso de redes móveis, tais como 3G e 4G. Porém essas redes móveis têm como foco aparelhos como *laptops*, *smartphones* e *tablets*. Em um ambiente na qual a IoT é amplamente implementada, haverá chips, sensores, dispositivos conectados por toda área, obrigando assim uma otimização das redes móveis atuais. É onde entra a 5G, um recurso que melhorará claramente a Internet móvel atual, evitando gargalos na rede e proporcionando maior velocidade do mesmo.

Sistema de Controle: Não adianta ter dispositivos que se conectam uns com os outros se não houver um controle adequado de seus dados, tais informações precisam ser tratadas, sejam em serviço nas nuvens ou grandes servidores, um sistema de controle é de grande importância quando se trata de IoT pois é através dele que será possível utilizar de forma adequada tudo o que a Internet das Coisas tem a oferecer.

6.1.1. Visão e Motivação

A IoT é um contexto que tem como objetivo desenvolver um caminho entre acontecimento do mundo real e as representações no ambiente digital, com o intuito de integrar ambos os paradigmas, obtendo assim benefícios ao seu uso. Atualmente existe muitos meios para realizar a captura dessas "coisas", desde códigos simples até os mais sofisticados e complexas tecnologias [Atzori et al. 2010].

Redes de sensores são atualmente muito estudados quando se trata de IoT, pois são classificadas como *Objetos Inteligentes,* ou seja, efetuam comunicações e operações computacionais entre si. Sensores é uma área muito estudada fora de ambientes acadêmicos, havendo aplicações em diversas áreas, essas redes de sensores são

agrupadas por um conjunto de nós, onde tais nós possuem funcionalidades diversas como medir temperaturas, ruídos, humidade, luminosidade, entre outros. Possuem como propósito coletar informações importantes e encaminhar para um ou mais nós específicos, que chamamos de *estações-base*, que funciona como ponte entre a rede de sensores com o ambiente externo, em casos específicos alguns nós possuem poder de processamento, assim sendo possível efetuar operações em cima de informações coletadas na rede.

6.1.2. Aplicações Existentes

Com o grande crescimento da Internet aliado a um eficiente processo tecnológico, é possível aplicar a IoT em diversas áreas, sejam em indústrias, serviços, produtos, entre outros. Uma área de aplicação que mais favorece é a melhoria na qualidade de vida das pessoas, usando a IoT para desenvolver edifícios e casas inteligentes [Vermesan e Friess, 2013].

No segmento de casas inteligentes devemos ressaltar um conjunto de controle ou gestão de eficiência em consumo de energia, onde podemos controlar aqueles aparelhos que estão consumindo energia em excesso, efetuando uma gestão dos aparelhos, sejam em ar-condicionado, termostatos ou segurança. Um exemplo seria um termostato que tem a capacidade de regular a temperatura de um ambiente de acordo com a presença de alguma pessoa no local, ajustando a temperatura de acordo com o critério definido pelo usuário [Vermesan e Friess, 2013].

Já na área de saúde podemos destacar aplicações que monitoram o estado de pacientes dentro de um hospital, emitindo algum sinal para os médicos quando algum paciente tem queda de pressão, ritmo cardíaco alterado, algumas anomalias no estado de saúde do paciente.

Em áreas agrícolas a IoT possibilita um monitoramento de suas lavouras sejam com o monitoramento do solo, umidade no ambiente, condições climáticas, detectando áreas que necessita de irrigação, detecção de níveis de agrotóxicos usados na lavoura, deste modo melhorando a qualidade e produção daquele produto, além de gerar menor custo ao produtor [Vermesan e Friess, 2013].

Em ambientes externos a IoT também pode contribuir, efetuando a prevenção de certos problemas, como monitoramento de combustão de gases, visualizar pontos com maior risco de incêndios, controlar o nível de emissão de poluição no ar, efetuar a prevenção de deslizamento de terra através do nível de humidade no solo, antecipar a detecção de terremotos através de sensores, [Vermesan e Friess, 2013].

A Internet das Coisas possibilita uma coleção de vantagens em sua utilização podendo ser usada em pequenas aplicações ou até mesmo ser aplicadas no conceito de cidades inteligentes.

6.1.3. Arquitetura da Internet das Coisas

A IoT possui um conceito de arquitetura composto por cinco Camadas: negócio, aplicação, processamento, transporte e percepção. O autor Miao et al. (2010) destaca

que a Internet das Coisas se difere da Internet, pois na IoT sua estrutura necessita ser gerenciada e operada assim, pensando nisso, a arquitetura da IoT foi desenvolvida com base no TMN *Telecommunications Management Network*.

Serão listadas e explicadas as respectivas camadas da arquitetura da Internet das coisas.

- **Camada de Negócio**: está camada atua como um gerente da IoT pois é responsável por efetuar questões administrativas com privacidade de usuários.
- **Camada de Aplicação:** está camada fica a cargo de desenvolver aplicações baseadas em informações analisadas em camadas anteriores
- **Camada de Processamento:** está camada fica a cargo do armazenamento, análise e processamento de dados encaminhados através da camada de transporte.
- **Camada de Transporte:** está camada fica a cargo de transmitir dados processados "informações", recebidas por meios de conexões como, *Wi-Fi*, infravermelho, 3G, rádio frequências, entre outros. Protocolos de endereçamento também são encontrados nesta camada.
- **Camada de Percepção:** está camada fica a cargo de efetuar a coleta de dados, tais informações são adquiridas através de atuadores, sensores, etc. Também é nessa camada que informações vão ser convertidas em sinais para ser transmitida na rede.

6.1.4. Computação nas Nuvens e a IoT

Computação em nuvem (*Cloud Computing*) idealizou um modelo de computação distribuída onde se tem recursos computacionais (hardware, plataformas de desenvolvimento, armazenamento e comunicação), que são disponibilizados e virtualizados como serviços reconfiguráveis. Muitas vezes são recursos pagos para poderem ser utilizados e geralmente por grandes centros de dados na internet, deste modo oferecendo suporte para os mais diversos serviços de armazenamento e compartilhamento de dados, independentemente do local que se encontra é possível acessar seus dados, bastando estar conectado à Internet. [Armburst et al., 2010]

O conceito em nuvem tem como objetivo ser global e prover serviços variados, que podem ir desde um usuário final que hospeda pequenos documentos de dados pessoais na Internet, até grandes empresas que terceirizam toda a infraestrutura da TI (*Tecnologia da Informação*) para outras empresas.

Computação em Nuvem é uma tecnologia que disponibiliza recursos computacionais virtualizados através da Internet, utilizando computadores e servidores que compartilham informações. Tal estrutura é capaz de integrar grandes redes a diversos tipos de sensores, a *Cloud Computing* apresenta algumas soluções para problemas denominados de *Sensor-Cloud* (*nuvem de sensores virtuais*), onde essa integração visa solucionar problemas de distribuição e armazenamento de dados adquiridos através de sensores, desta forma permitindo processar informações de várias Redes de Sensores Sem Fio (RSSF), melhorando assim o compartilhamento de informações em grande escala. [Ansari et al., 2013]

Uma *Sensor-Cloud* motiva a necessidade de uma estrutura de RSSF, com o intuito de melhorar as operações e manutenções do sistema, permitindo fluxos de comunicação direta no meio de dispositivos heterogêneos.

Redes de Sensores Sem Fio é visualizada como conceitos e elementos da IoT, que é baseada na junção digital com o meio externo, desta forma tais tipos de estrutura é vista com uma concepção promissora de cidades inteligentes, onde a computação em nuvem disponibiliza soluções para gestão da *Internet das Coisas*, sendo então uma alavanca para implementação e o bom funcionamento do futuro da IoT.

6.1.5. Exemplos

Nesta sessão são apresentados dois exemplos de aparelhos que fazem uso da IoT para benefício de seus usuários. O primeiro é o Nike FuelBand SE (Figura 6.1) uma pulseira conectada a Internet e através dela é possível registrar atividades físicas do usuário, como: distância percorrida, velocidade alcançada, contador de passos, verificador de calorias gastas, etc.

Figura 6.1. Pulseira Nike conectada a Internet

Outro exemplo é o Nest Learning Thermostat (Figura 6.2), um produto baseado na Internet das Coisas. Este produto é basicamente um termostato inteligente cuja função é aprender quais níveis de temperatura deve estar em determinada hora do dia, ou seja, o termostato vai saber que temperatura deve estar durante a noite, pela manhã, ou de tarde, o produto é controlado através de um smartphone, onde o usuário consegue controlar o termostato através da internet.

Figura 6.2. Termostato Inteligente conectado a Internet

6.2. Arduino

A plataforma Arduino foi criada em 2005 por Massimo Banzi et. al. (2005) com propósito de se tornar uma plataforma livre e de baixo custo para ser utilizada por artistas, designers, hobbistas, para fins educacionais e qualquer pessoa interessada em

criar objetos ou ambientes interativos [Arduino, 2016a] [Perez, 2013]. O Arduino é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software flexíveis e fáceis de usar. [Arduino, 2016b]

Este tópico apresentará uma visão geral sobre a plataforma Arduino, seus componentes e os primeiros passos para iniciar o desenvolvimento utilizando esta plataforma.

6.2.1. Introdução

Arduino pode ser utilizado para coisas tão simples como piscar um Led, ou algo bem mais elaborado, como interagir com um smartphone entre outros [Boxall, 2013]. É possível ainda utilizar de sensores para sentir o ambiente que o cerca, de forma a interagir com o ambiente, controlar luzes, motores ou outros atuadores. Além disso, é possível criar projetos autônomos utilizando Arduino, ou projetos que se comunicam com computadores para realizar tarefas mais complexas, utilizando softwares específicos. [Arduino, 2016b]

O Arduino é formado por dois componentes, um de hardware e o outro de software. O hardware é uma placa de prototipagem onde são construídos os projetos. A placa do Arduino pode ser construída de forma caseira, adquirindo-se os componentes e montando manualmente, ou a placa pode ser adquirida já montada. O projeto para montagem, que pode ser baixado gratuitamente, está disponível sob licença *open-source*, de forma que é possível adaptá-lo para cada propósito [Arduino, 2016b]. A Figura 6.3 mostra o Arduino utilizado neste texto, o Arduino Leonardo.

Figure 6.3. Arduino Leonardo

O software é uma IDE, a ser instalada no computador para a programação do controlador. A IDE é conhecida como sketch e é onde toda a programação é feita, ou seja, como o controlador se comportará. O upload da programação para o controlador é feito através de comunicação serial. O micro controlador do Arduino é programado com a linguagem de programação Arduino, criada a partir da linguagem Wiring. Já o ambiente de programação é baseado no ambiente Processing. [Souza, 2013]

6.2.2. Instalação

A programação do Arduino é feita através de ambiente de desenvolvimento integrado (IDE) próprio, que pode ser baixado do link: https://www.arduino.cc/en/Main/Software. Neste link é possível fazer o download do arquivo instalador ou do pacote, recomendamos o instalador, pois já permite a instalação de todos os componentes necessários, incluindo os drivers. [Arduino, 2016c]

Ao efetuar o *download* do arquivo, a instalação é simples, basta seguir os passos a seguir:

1. Execute o arquivo baixado e aceite a Licença de Uso (Figura 6.4);
2. Nas Opções de Instalação marque todas as opções (Figura 6.5);
3. Escolha o local de instalação e clique em Instalar (Figuras 6.6 e 6.7);
4. Após a instalação, serão exibidas duas mensagens de segurança do Windows solicitado a instalação de dois drivers, permita que sejam instalados (Figuras 6.8 e 6.9).

Figura 6.4. Licença de Uso

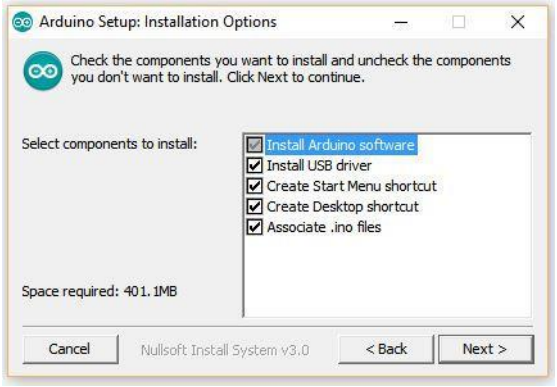


Figura 6.5. Opções de Instalação

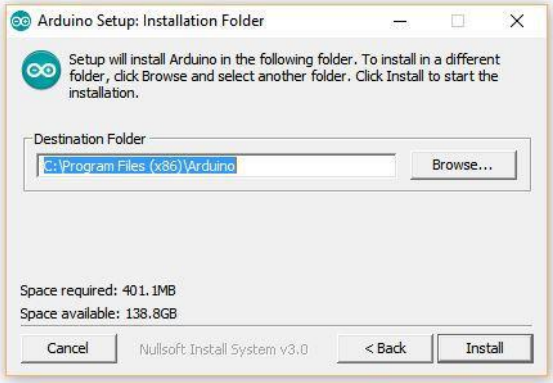


Figura 6.6. Local de Instalação

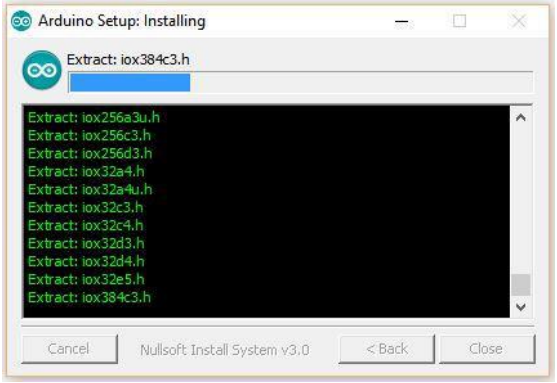


Figura 6.7. Instalação

Figura 6.8. Instalação de Driver

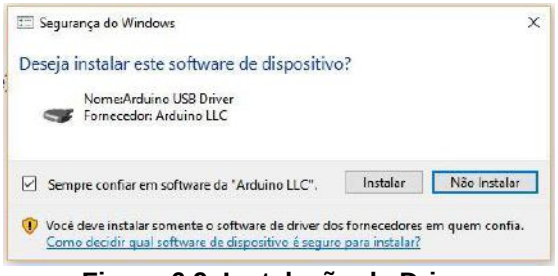


Figura 6.9. Instalação de Driver

Assim que a instalação for concluída, você deve conectar a placa do Arduino no computador e aguardar o reconhecimento e instalação do driver.

6.2.3. Programação no Arduino

A programação do Arduino é feita através do ambiente de desenvolvimento Arduino (Figura 6.10). O código mostrado na Figura 2.9 representa o mínimo para que um programa possa ser compilado [Robot@escola, 2016]:

- O "void setup()" é utilizado para a inicialização das variáveis, modos de pinas, onde é definida a velocidade de transmissão serial, dentre outras funções.
- O "void loop()" é onde se cria um ciclo de repetição para as instruções que devem ser executadas repetidamente.

Além dos dois elementos principais previamente destacados, o código do Arduino é constituído de mais 3 elementos:

- As bibliotecas, incluídas no início do programa, são constituídas de funções pré-feitas pela comunidade de programadores ou pelos criadores da plataforma.
- As variáveis são declaradas de forma global, após a importação das bibliotecas.
- As funções, criadas pelo programador, seguem às declarações de variáveis.

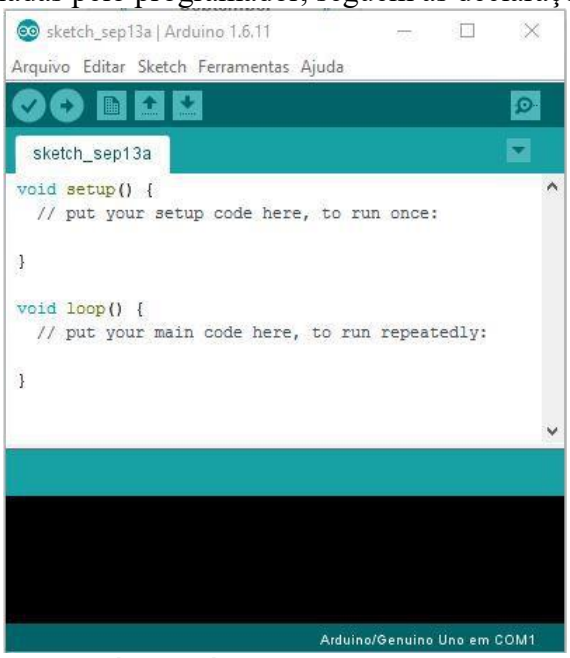


Figura 6.10. Código básico de programação do Arduino

Para carregar e iniciar a programação no Arduino, basta clicar no botão upload, no Arduino IDE. O software do Arduino comanda automaticamente um reset na placa, lançando um *bootloader* – que é responsável por receber, armazenar, e iniciar o novo sketch. [Aloi, 2013]

Como exemplo, o código mostrado na Figura 6.11, controla o Led do pino D13 da placa do Arduino.

Figura 6.11. Código para controle do LED do pino D13 (Laboratório de Garagem, 2014)

6.2.4. Instalação e uso do módulo Ethernet Shield

O módulo Ethernet Shield é utilizado para conectar o Arduino a uma rede Ethernet, permitindo criar projetos que envolvam o envio e o recebimento de informações através da rede ou da internet. A Instalação do módulo é feita de forma simples, basta conectar o módulo Ethernet Shield à plataforma Arduino, como mostra a Figura 6.12 [Arduino e cia, 2013].

Figura 6.12. Encaixe do Módulo Ethernet [ARDUINO E CIA, 2013]

Após encaixar o Ethernet Shield à placa do Arduino, deve-se conectar o cabo de rede. Com o shield ethernet devidamente encaixado na placa do Arduino, basta ligar o cabo de rede. Na parte superior, temos os leds de status, que mostram o funcionamento do módulo e o status de conexão à rede, como mostra a Figura 6.13. [Arduino e cia, 2013]

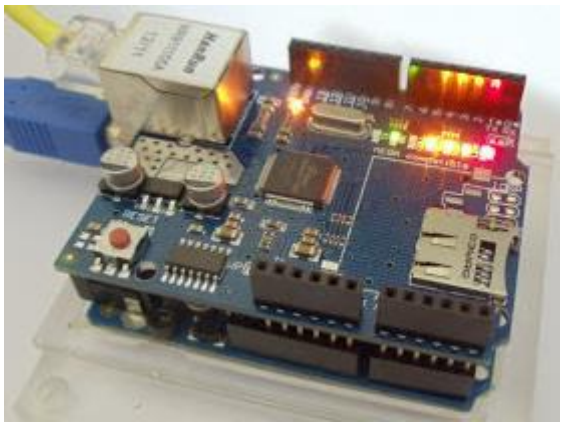


Figura 6.13. Módulo Ethernet conectado [Arduino e cia, 2013]

Como o dispositivo tem uso simplificado, não é necessário se preocupar com esquemas de sinalização de redes ethernet, uma vez que todo o controle e conexão é feito através de funções da biblioteca. Uma vez conectado, pode-se ler ou escrever dados através da conexão, de forma semelhante à uma conexão serial convencional. [Cândido, 2013]

Para testar a placa, será apresentado um projeto simples, que configura o endereço IP para valores apresentados na Figura 6.14.

Endereço IP	192.168.0.100
Gateway	192.168.0.1
Máscara	255.255.255.0

Figura 6.14. Endereços a serem configurados no módulo Ethernet [Arduino e cia, 2013]

O código utilizado para teste do Módulo Ethernet é mostrado na Figura 6.15. Como exemplo, está apenas sendo configurado os endereços de IP, Gateway e Máscara do módulo, para os valores mostrados na Figura 6.14. É importante salientar que os endereços IPs são separados por vírgula, ao invés do ponto, como é normal. O código da Figura 6.15 deve ser carregado para o Arduino.

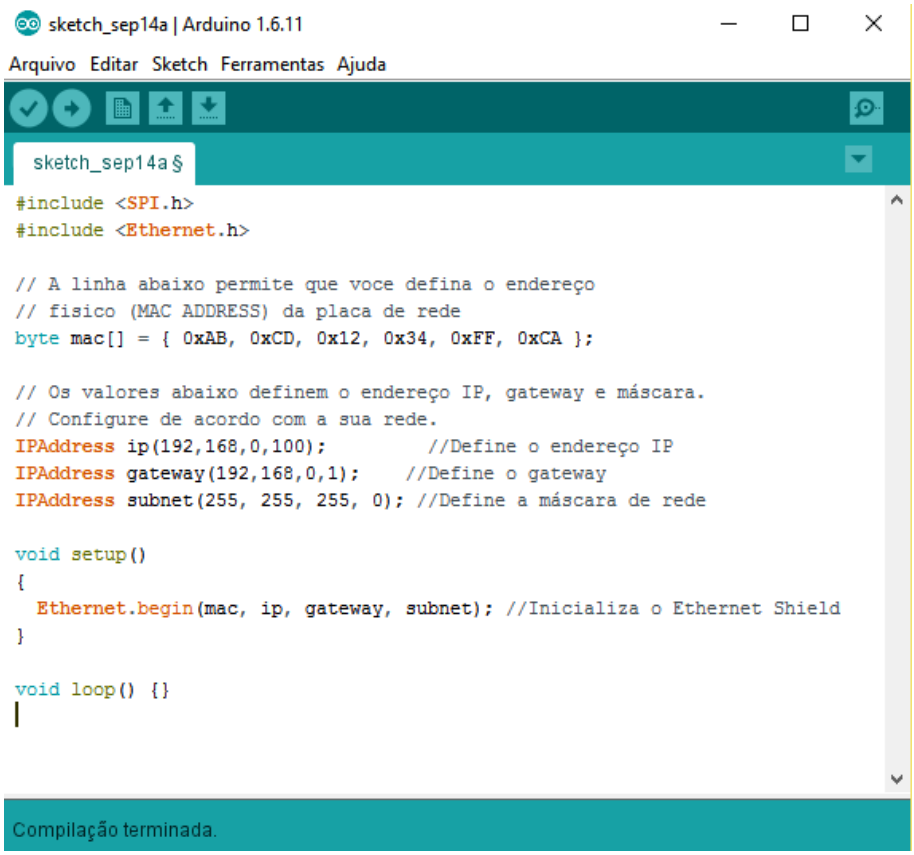


Figura 6.15. Programa Exemplo de comunicação com o Módulo Ethernet [Arduino e cia, 2013]

Para testar se o Módulo foi configurado corretamente, utilize o comando ping. Abra o *prompt* de comando do Windows, e digite o comando: ping 192.168.0.100 (o endereço que utilizamos para configurar o módulo). Caso o módulo tenha sido configurado com sucesso, o resultado será como o mostrado na Figura 6.16.

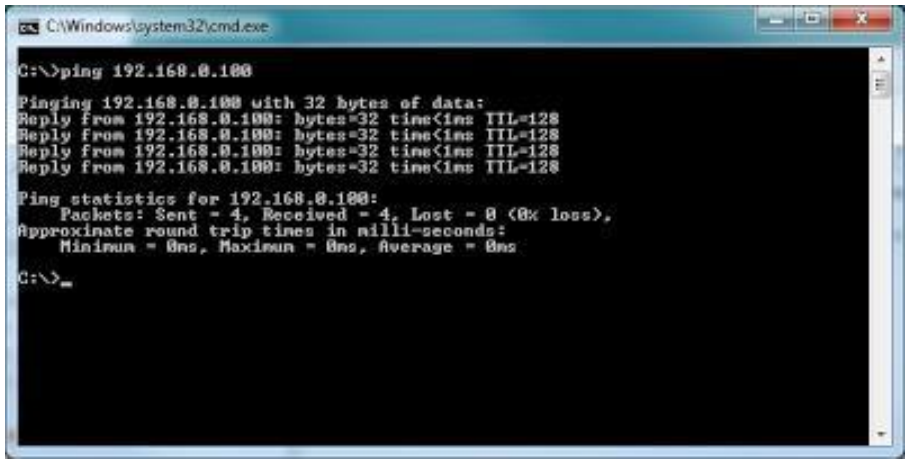


Figura 6.16. Resultado do comando ping [Arduino e cia, 2013]

6.2.5. Sensores e Atuadores

De forma a permitir maior ação do Arduino, podem ser acoplados a ele sensores e atuadores. Os sensores são conversores de grandezas físicas em sinais elétricos, já os atuadores, convertem energia elétrica, hidráulica ou pneumática em energia mecânica, permitindo gerar movimento. [Automação e robótica, 2012]

Os sensores podem ser classificados como:

- Sensores externos: que fazem observação do ambiente exterior ao Arduino. Pode-se citar sensores de contato, de proximidade, de força, de distância, de laser, de ultrassom, de infravermelhos e sensores químicos. [Ribeiro, 2004]
- Sensores internos: fornecem informação sobre os parâmetros internos do projeto de Arduino, como a velocidade ou sentido de rotação de um motor, ou o ângulo de uma junta. Potenciômetros, codificadores e os sensores inerciais (incluindo acelerômetros, giroscópios, inclinômetros e bússolas), são exemplos de sensores internos. [Automação e robótica, 2012]

Os sensores também podem ser classificados de acordo com a forma que a energia é envolvida no processo de sensoriamento. [Automação e robótica, 2012]

- Sensores ativos: como sensores laser, ultrassom e de contato, realizam o sensoriamento através da emissão de energia para o ambiente ou por modificarem o ambiente.
- Sensores passivos: como os sensores óticos, recebem energia do ambiente [Ribeiro, 2004].

Ainda existem os sensores de distância, como o laser ou o ultrassom, sensores de posicionamento absoluto, por exemplo sistemas de GPS, sensores ambientais (de temperatura, de umidade, etc.) e inerciais (que indicam a aceleração ou a velocidade) [Ribeiro, 2004].

Os Atuadores realizam a conversão da energia elétrica, hidráulica ou pneumática em energia mecânica. É possível classificar os atuadores de acordo com o tipo de energia que utiliza [Ribeiro, 2004]:

- Atuadores Hidráulicos: utilizam um fluido à pressão para movimentar o braço. Possuem baixa precisão, mas têm grande potência e velocidade, por isso são utilizados em robô que operam grandes cargas.
- Atuadores Pneumáticos: utilizam um gás à pressão para movimentar o braço. Possuem baixa precisão, e são utilizados em tarefas do tipo pega-e-coloca em robôs de pequeno porte.
- Atuadores Eletromagnéticos: motores elétricos (de passo, servos, Corrente Continua ou Corrente Alternada) ou músculos artificiais, usados em robôs de pequeno e médio porte.

6.3. Node.js

A Internet das Coisas necessita, entre outros recursos, de uma comunicação cliente servidor em tempo real no sentido de tornar eficaz a troca de informações entre os diversos dispositivos conectados. Existem diversas tecnologias que podem cumprir com este papel, todavia algumas possuem melhor desempenho e são mais adequadas para uma situação especifica. Também podemos escolher a tecnologia com base em sua simplicidade e compatibilidade com os recursos disponíveis atualmente [Pereira, 2014].

A maior parte dos sistemas web desenvolvidos atualmente contam com uma arquitetura bloqueante (*Blocking-Thread*), isto é, enquanto um processo é executado os demais ficam em uma fila aguardando para serem executados [Pereira, 2014]. Logo percebemos que este tipo de arquitetura pode trazer lentidão ao sistema e exigir diversos recursos computacionais para manter a fila. Tecnologias que apresentam tal arquitetura não são recomendadas, uma vez que um dos objetivos da Internet das Coisas é uma comunicação rápida entre sensores e elementos de processamento.

O Node.js nasceu em 2009, apresentando como proposta ser uma arquitetura não bloqueante (*non-blocking thread*). O Node.js aproveita ao máximo os recursos disponíveis e garante uma boa performance em sistemas que trabalham com grande carga de processamento. É uma plataforma escalável de baixo nível, onde o programador trabalha com protocolos de rede e internet ou usa bibliotecas especificas para uma determinada tarefa [Pereira, 2014].

A linguagem de programação utilizada no Node.js é o Javascript, pois sua construção foi baseada na *engine* Javascript V8 [Node.js, 2016].

6.3.1. Instalação e Configuração

O Node.js está disponível e pode ser baixado gratuitamente no endereço <https://nodejs.org/en/download/>, seu processo de instalação é simples e pode ser feito em diversas plataformas, conforme mostra a Figura 6.17. A configuração aqui apresentada foi feita em um ambiente com sistema operacional Windows, mas é semelhante nas demais plataformas.

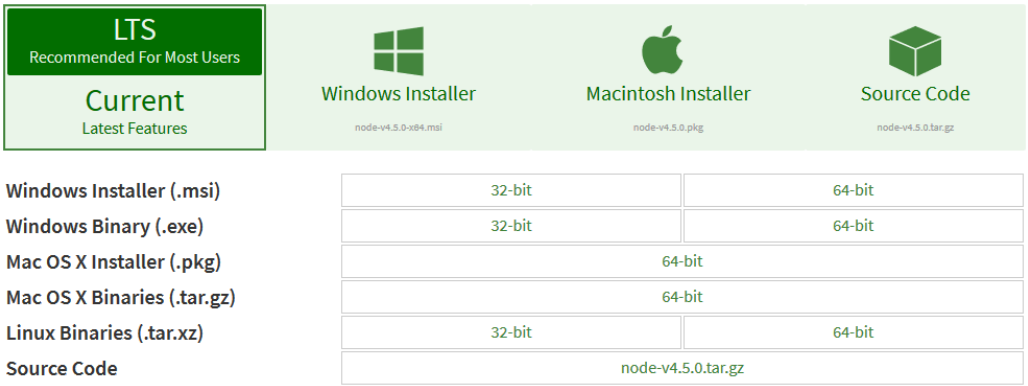


Figura 6.17. Tela de download do Node.js [Node.js, 2016]

Após efetuar o *download* da versão desejada, é necessário clicar no executável e dar início ao processo de instalação, exceto quando se tratar do sistema operacional

Linux. No processo de instalação é necessário apenas concordar com os termos e aguardar até que seja finalizado.

Após a conclusão da instalação podemos verificar se o Node.js está funcionando perfeitamente, para isto é necessário apenas abrir o *prompt* e executar o seguinte comando: `node -v`, conforme mostra a Figura 6.18, onde vemos que foi instalada a versão 4.5.0 do Node.js.

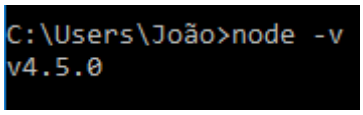


Figura 6.18. `Prompt` **de comando mostrando a versão do Node.js instalada.**

Para ter certeza que o Node.js está funcionando perfeitamente ainda podemos testar nosso primeiro programa que mostra uma mensagem *Hello Word*. Para criar e executar o programa basta digitar o comando: `node`, que nos permite executar Javascript no terminal e fornecer o seguinte comando: `console.log("Hello Word");` depois da execução do programa aparecerá a mensagem, conforme mostra a imagem 6.19.

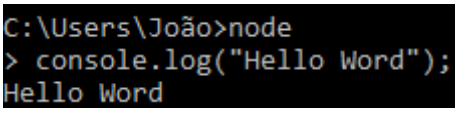


Figura 6.19. Testando o primeiro programa no prompt de comando.

O Node.js conta com um gerenciador de pacotes, ele se chama NPM (*Node Package Manager*) [Pereira, 2014]. Este gerenciador de pacotes possui o mesmo objetivo dos demais gerenciadores que estão presentes em outras tecnologias de desenvolvimento, isto é, gerenciar pacotes que são necessários para o desenvolvimento e execução de uma determinada aplicação. A seguir são listados e descritos alguns comandos do NPM.

- `npm install nomeDoMódulo`: instala um determinado módulo no projeto.
- `npm install nomeDoMódulo –save`: instala um determinado módulo e atualiza o package.json na lista de dependências da aplicação.
- `npm list`: lista todos os módulos do projeto.
- `npm –v`: mostra a versão atualmente instalada do npm.
- `npm remove nomeDoMódulo`. Remove um módulo do projeto.

Existem diversos outros comandos, a documentação completa do NPM pode ser obtida em <https://docs.npmjs.com/>.

Existe uma infinidade de módulos disponíveis em <https://www.npmjs.com/>, cada um para uma determinada tarefa. É interessante que o programador mantenha um arquivo com o nome e a versão de cada módulo que o sistema necessitar, tanto para ele, quanto para outro desenvolvedor que futuramente estiver realizando alguma manutenção na aplicação. No Node.js usa-se o arquivo `package.json`.

O arquivo `package.json` é formado principalmente por dois atributos o nome e o versão dos pacotes que serão instalados com a execução do seguinte comando: `npm install`. Um exemplo de um arquivo `package.json` é apresentado a seguir:

```
{
   “name”: “primeiro_projeto”,
   “version”: “0.0.1”,
   “description”: “um_projeto_qualquer”,
   “author”: “senhor_exemplo”,
   “private”: true,
   “dependencies”: {
      “módulo 1”: “0.1.2”,
      “módulo 2”: “1.1.2”
}
}
```

As dependências que foram descritas, módulo 1 e módulo 2, serão instalados com a execução do comando: `npm install`, conforme mencionado anteriormente. A criação deste arquivo pode ser feita manualmente ou com a ajuda de um assistente que pode ser acessado pelo prompt através da execução do seguinte comando: `npm init`, em seguida serão solicitadas algumas informações e ao fim do processo um arquivo package.json será gerado, conforme mostra a imagem 6.20.

```
C:\Users\João>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (João) PrimeiroProjeto
Sorry, name can no longer contain capital letters.
name: (João) primeiro_projeto
version: (1.0.0)
description: um projeto qualquer
entry point: (index.js)
test command:
git repository:
keywords: projeto qualquer meu
author: senhor_exemplo
license: (ISC)
About to write to C:\Users\João\package.json:

{
  "name": "primeiro_projeto",
  "version": "1.0.0",
  "description": "um projeto qualquer",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "projeto",
    "qualquer",
    "meu"
  ],
  "author": "senhor_exemplo",
  "license": "ISC"
}
```

Figura 6.20. Criação de um arquivo package.json através do assistente

6.3.2. Criando a primeira aplicação com Node.js

Agora que já aprendemos um pouco sobre Node.js, podemos criar nossa primeira aplicação Web.

Primeiramente vamos criar um arquivo chamado: *server.js*, este será o nosso servidor. Após a criação do arquivo, inicialmente vamos declarar o módulo http que é nativo do Node.js, através da seguinte instrução:

```
http = require('http');
```

Após a declaração do módulo http, vamos criar uma função que é responsável por levantar o servidor e colocar seu call-back em funcionamento. O callback é representado pela função: function(request, response) e é executado toda vez que a aplicação recebe uma requisição.

```
var servidor=http.createServer(function(request, response)){
... });
```

A função declarada é chamada de *Event Loop*, ou seja, a cada requisição ele será acionada e irá enviar uma resposta ao usuário. Vamos enviar uma mensagem para o

usuário conforme o endereço acessado, ou seja, vamos criar rotas para o usuário acessar através da url. As rotas e o envio de uma mensagem em uma página da web pode ser feitas da seguinte forma:

```
var servidor = http.createServer(function(request, response){
  response.writeHead(200, {"Content-Type": "text/html"});
  if(request.url == "/")
    response.write("<h1>Página principal do meu site</h1>");
  else if(request.url == "/contato")
    response.write("<h1>e-mail: exemplo@exemplo.com !!</h1>");
  else
    response.write("<h1>Página não encontrada!!</h1>");
    response.end();
});
```

O método `http.writeHead` constrói um novo cliente HTTP e código 200 refere-se é um código de sucesso da requisição HTTP, existem diversos outros códigos de status que podem ser informados neste método. Foram criadas diversas condições e dentro de cada uma é exibida uma mensagem ao usuário. O método request.url retorna o endereço que o usuário acessou.

A aplicação está quase terminada, falta apenas a criação do método `listen` que traz a porta que o servidor está sendo executado. Este método pode ser criado através das seguintes instruções:

```
servidor.listen(3000, function() {
     console.log("O servidor está em execução"); });
```

O servidor será executado na porta 3000, conforme mostra o código. Agora a aplicação está terminada e para ser executada são necessários os seguintes passos:

1. Navegar até o diretório da aplicação;
2. Executar o comando: `node nome_servidor.js;`
3. Abrir o navegador e acessar o endereço: http://localhost:3000/.

Após a execução destes passos é possível navegar entre as diversas rotas criadas no servidor.

A simplicidade e praticidade do Node.js tornam esta tecnologia ideal para o desenvolvimento de aplicações da Internet das Coisas.

6.4. Desenvolvimento de aplicação prática

Nesta sessão será apresentado um sistema de controle de leds via página Web com Arduino. A aplicação consiste no controle de leds montados em um circuito conectado a uma placa arduíno e controlado por uma página web hospedada em um servidor criado com o framework Node.js

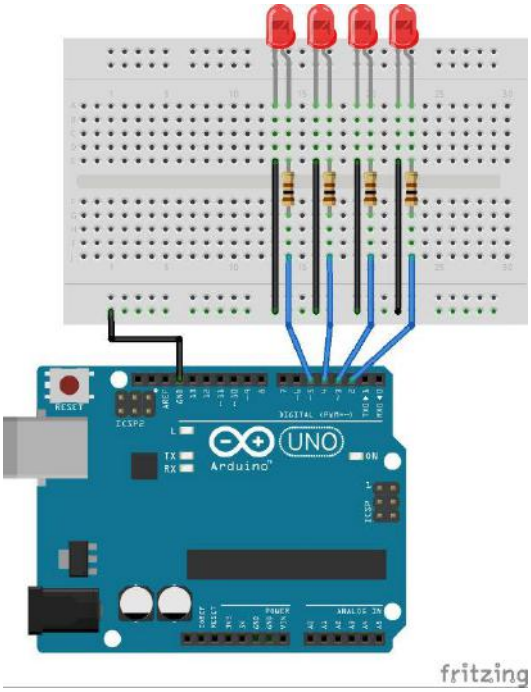
6.4.1. Montagem do Circuito

Para a montagem do circuito será necessário a utilização dos seguintes materiais:

- Uma placa arduíno
- Uma protoboard (placa onde é montado o circuito)

- 4 leds
- 9 fios
- 4 resistores de 300k

O circuito montado é apresentado na Figura 6.21.



6.21. Circuito montado [PedroHS, 2015]

6.4.2. Implantação do Código no Arduíno

Para o funcionamento do sistema é necessário que seja importado para o arduíno um firmware que permite sua comunicação com um computador servidor, no caso o localhost, ou seja, a máquina que possui o servidor Node.js instalado e ficará ligada ao arduíno através de um cabo USB.

O protocolo a ser utilizado será o Firmata que é uma biblioteca de comunicação do hardware com o computador servidor. Ele permitirá que o hardware seja controlado pelo software do servidor. Então para a aplicação é necessário a instalação do Firmata no Arduino como um firmware e assim ele estabelecerá a comunicação entre hardware e software no computador servidor para que este controle o Arduino [Arduino, 2016d] [Firmata, 2016].

Para a instalação é necessário abrir o ambiente Arduino IDE (Figura 6.22)

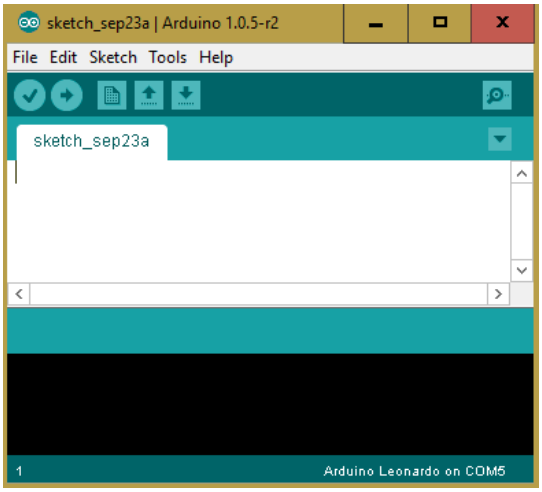


Figura 6.22. Tela inicial do Arduino IDE

Depois vá em Examples > Firmata > StandardFirmata (Figura 6.23_

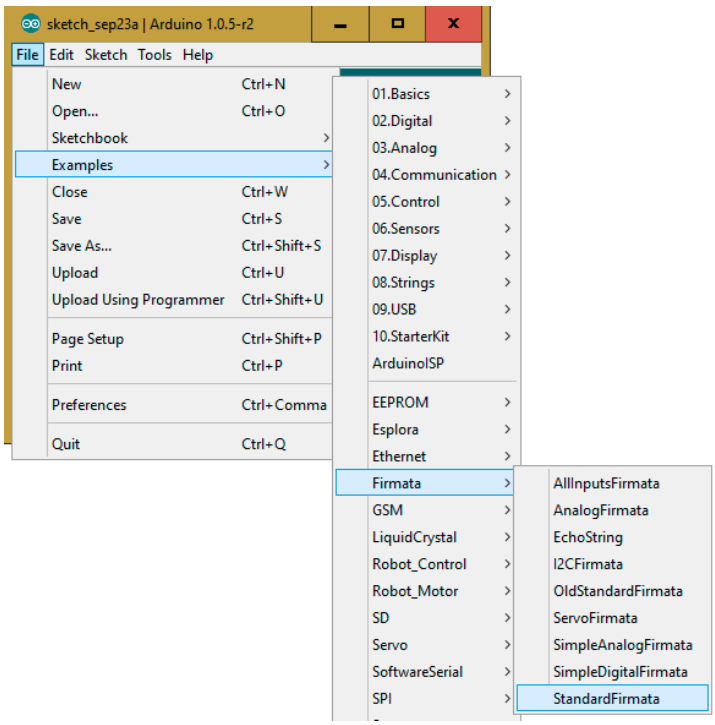


Figura 6.23. Selecionando o protocolo StandardFirmata

Ao selecionar o programa StandardFirmata ele será carregado na área de edição de código do Arduino IDE (Figura 6.23).

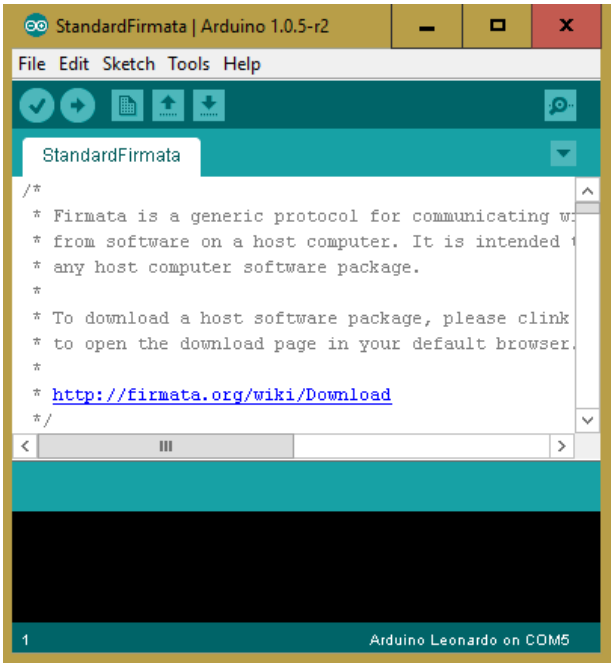


Figura 6.24. Código do StandardFirmata no ambiente do Arduino IDE

Antes do código ser enviado para o arduino é necessário executar a verificação do código com o ícone  . Esta operação evitará que o código com erros seja transferido para a placa arduino o que evita que este venha a danificá-la. Para que o código seja carregado no arduíno o botão. Depois de verificar o código é necessário realizar o upload para a placa arduíno através do ícone  .

Após a transferência do protocolo para o arduino será exibido no rodape do Arduino IDE uma mensagem de confirmação que o software foi transferido corretamente.

A partir da transferência a comunicação entre o computador e o arduino poderá ser feita permitindo que uma página web através do servidor Node.js controle os leds que estão montados no circuito.

6.4.1. Criação da Aplicação Javascript

A aplicação será construída em Javascript utilizando o servidor Node.js e o framework de código aberto Johnny Five que permite controlar o harware utilizando Javascript. Para instalá-lo é necessário instalar o Node.js e NPM (como descrito na sessão 6.2). Em seguida criar a pasta onde será colocado o projeto e via prompt ela deverá ser selecionada. Para instalação do framework Johnny Five na pasta do projeto o seguinte comando deverá ser executado no prompt de comando: `npm install johnny-five` (Figura 6.25).

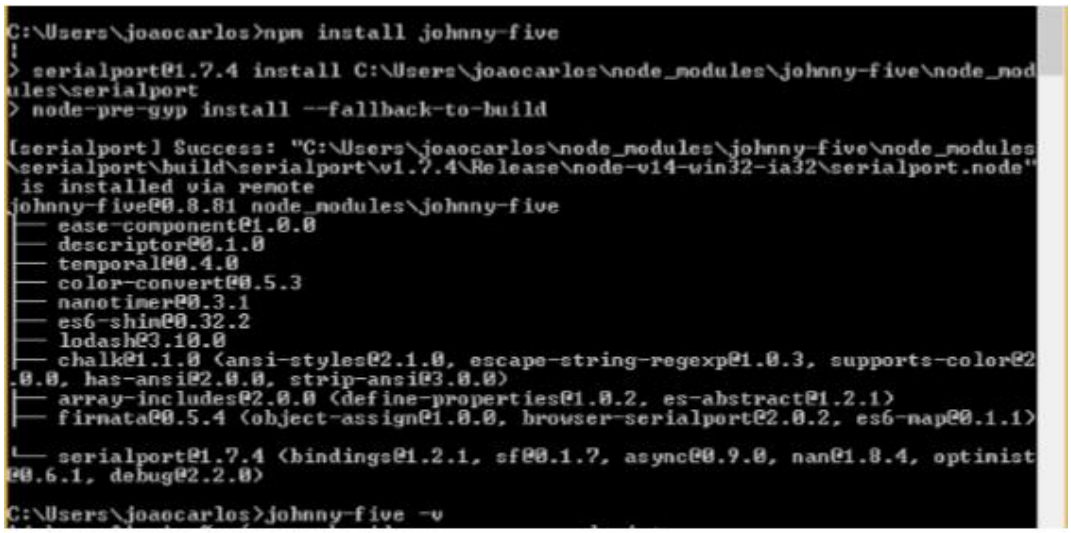


Figura 6.25. Instalação do johnny-five no terminal do Windows

Após a instalação do framework Johnny Five é necessário criar o servidor que atenderá as requisições feitas pelo navegador. Este servidor será chamado **app.js** e deverá ser gravado na pasta do projeto. Na Figura 6.26 é apresentado as definições das bibliotecas que serão utilizadas e variáveis e na Tabela 6.1 cada linha do código é comentada.

```
1 var http = require('http').createServer(servidor);
2 var fs = require('fs');
3 var five = require('johnny-five');
4 var arduino = new five.Board();
5 var led0, led1, led2, led3;
```

Figura 6.26. Importação das bibliotecas e definição das variáveis

Tabela 6.1. Comentários em relação ao código da Figura 6.26

Linha	Comentário
1	O módulo com o protocolo HTTP é importado (o módulo http é nativo do Node.js) e a função createServer é responsável por deixar o servidor passível de receber requisições. O método recebe como parâmetro o nome da função que será responsável por iniciar o servidor.
2	O módulo fs (File System) permite a leitura de arquivos externos.
3	O módulo johnny-five permite controlar a placa arduino através do Javascript
4	Criação do objeto arduino (representação em software da placa arduino) através do método five.Board()
5	Variáveis que irão armazenar a representação em software dos Leds ligados à placa arduíno.

Na linha 4 da Figura 6.26 foi criado o objeto arduino que é a representação em software da placa arduino através do método da biblioteca johnny-five `Board()`. Este método inicia a comunicação com a placa, conectando o servidor com ela e somente após a comunicação ser estabelecida é possível identificar os componentes da placa e executar a ligação destes com seus correspondentes em software. Na Figura 6.27 é apresentado o código da função que é executada quando a comunicação entre servidor e na Tabela 6.2 cada linha de código da Figura 6.27 é comentada.

```
 7 arduino.on('ready', function(){
 8     console.log("Pronto");
 9     led0 = new five.Led(2);
10     led1 = new five.Led(3);
11     led2 = new five.Led(4);
12     led3 = new five.Led(5);
13 });
```

Figura 6.27. Conexão com a placa arduino

Tabela 6.2. Comentários em relação ao código da Figura 6.27

Linha	Comentário
7	Quando a placa arduíno entra no estado de 'ready', ou seja, quando a comunicação entre hardware e servidor é estabelecida a função definida entre as linhas 8 e 12 é chamada
8	Uma mensagem é exibida no terminal de comando
9 a 12	Cada led é presentado em software através do método do framework johnny-five `Led()` que recebe como parâmetro a porta digital da placa arduíno na qual cada Led está conectado.

É necessário criar uma função que representa o servidor que irá atender as requisições do usuário. Na Figura 6.28 é apresentado o código do método servidor e na Tabela 6.3 cada linha de código é comentada.

```
15 function servidor(req, res){
16     var url = req.url;
17     if(url == '/'){
18         res.writeHead(200);
19         res.end(fs.readFileSync('view/index.html'));
20     } else if(url == '/led0'){
21         res.writeHead(302, {'Location': '/'});
22         res.end();
23         led0.toggle();
24     }else if(url == '/led1'){
25     //...
26     } else{
27         res.writeHead(200);
28         res.end("<h1> Erro 404 </h1>");
29     }
30 };
```

Figura 6.28. Método servidor

Tabela 6.2. Comentários em relação ao código da Figura 6.28

Linha	Comentário
15	O método recebe como parâmetro os objetos de requisição e resposta (req e res respectivamente). O objeto requisição possui as informações que são enviadas ao servidor e o objeto resposta será a mensagem que retornará ao cliente (solicitante da requisição)
16	A url da requisição é lida
17, 20, 24	Verificação de qual url foi solicitada

18	O método res.writeHead constrói um novo cliente HTTP e código 200 refere-se é um código de sucesso da requisição HTTP
19, 22, 28	Finalização da conexão com o cliente, podendo enviar alguma informação. Na linha 19 estamos redirecionando o cliente para a página view/index.html e na linha 28 enviamos uma mensagem em HTML
21	O código 302 indica um redirecionamento para, neste caso, a página raiz, ou seja a index
23	O método `toggle()` muda o estado do Led, se ele estiver aceso ele apagará e vice versa.

A aplicação está quase terminada, falta apenas a criação do método `listen` que traz a porta que o servidor está sendo executado (Figura 6.29). Função já descrita na sessão 6.3.2.

```
32 http.listen(3000, function(){
33     console.log("Servidor on-line!");
34 });
```

Figura 6.29. Método listen

Agora que temos o servidor pronto é necessário a criação da página na qual o cliente terá acesso e poderá mudar o estado dos leds. O arquivo se chamará index.html (Figura 6.30) e ficará dentro da pasta view que será criada na pasta do projeto.

```
 1 <html>
 2   <head>
 3     <title>Aplicação IoT Leds</title>
 4   </head>
 5   <body>
 6     <ul>
 7       <li><a href="/led0">Led 01</a></li>
 8       <li><a href="/led1">Led 02</a></li>
 9       <li><a href="/led2">Led 03</a></li>
10       <li><a href="/led3">Led 04</a></li>
11     </ul>
12   </body>
13 </html>
```

Figura 6.30. Código HTML da página view/index.html

A aplicação agora poderá ser executada através da execução em prompt do comando `node app.js`.

Na Figura 6.31 podemos ver que a placa arduíno foi reconhecida como conectada à porta serial COM3 e a mensagem do status do servidor também foi mostrada. Agora precisamos acessar a view que foi criada, para isso é necessário acessar o endereço <localhost:3000> (Figura 6.32).

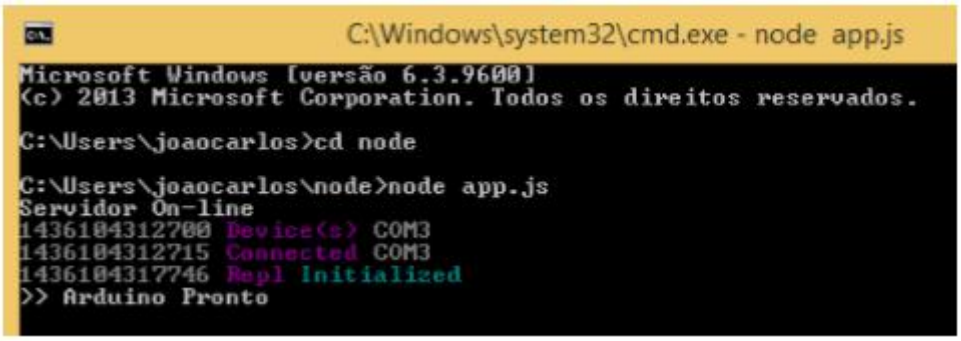


Figura 6.31. Execução da aplicação

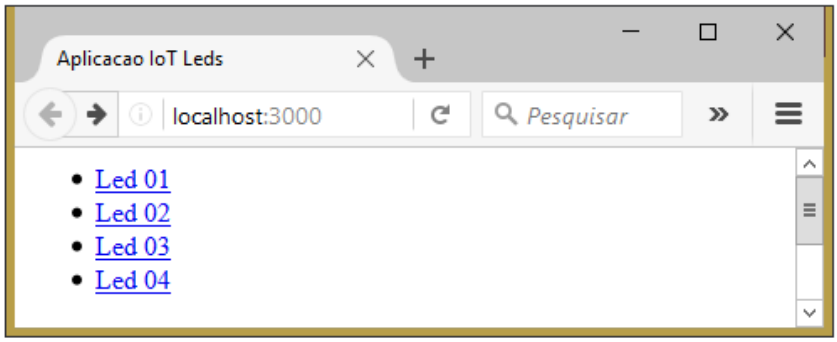


Figura 6.31. Renderização em navegador da página view/index.html

Ao clicar nos links da Figura 6.31 os estados de ligado/desligado dos Leds é alternado.

Referências

Aloi, R. (2013) "Guia para o Arduino Leonardo", Disponível em: http://renatoaloi.blogspot.com.br/2013/12/guia-para-o-arduino-leonardo.html Acesso em: Set. 2016

Ansari, W. S. et al. (2013) "A Survey on Cloud-Sensor: architecture, applications and approaches". In: J. Distrib. Sens. Networks, [S.l.], v.2013, p. 1–36.

Arduino (2016a), "*Credits*", Disponível em: https://www.arduino.cc/en/Main/Credits, Acesso em: Set. 2016

Arduino (2016b), "*Learning*", Disponível em: http://playground.arduino.cc/Portugues/HomePage, Acesso em: Set. 2016

Arduino (2016c), "*Install the Arduino Software (IDE) on Windows PCs*", Disponível em: https://www.arduino.cc/en/Guide/Windows, Acesso em: Set. 2016

Arduino (2016d), "*Firmata Library*", Disponível em: https://www.arduino.cc/en/reference/firmata, Acesso em: Set. 2016

Arduino e Cia, (2013), "Ethernet shield Wiznet W5100 - Parte 1", Disponível em: http://www.arduinoecia.com.br/2013/06/ethernet-shield-wiznet-w5100-parte-1.html. Acesso em: Set. 2016

Armbrust, M., Fox, A., Griffith, R., et al. (2010) “A View of Cloud Computing”. Communications of The ACM, Vol 53, No 4, p. 50-58.

Atzori, L., Iera, A., e Morabito, G. (2010) “The Internet of Things: A survey”. Computer Networks.

Automação e Robótica, (2012). “Sensores, Atuadores e Unidades de Controle”. Disponível em: http://automacaoerobotica.blogspot.com.br/2012/07/sensores-e-atuadores-aplicados-robotica.html, Acesso em: Set. 2016

Boxall, J. (2013). “*Arduino workshop: A Hands-On introduction with 65 projects*”. No Starch Press.

Candido, R., (2013). “Acendendo um LED via Internet com Arduino e o Ethernet Shield”. Disponível em: https://br.renatocandido.org/2013/09/acendendo-um-led-via-internet-com-arduino-e-o-ethernet-shield/. Acesso em: set. 2016

Firmata (2016) “Firmata”. Disponível em: http://firmata.org/wiki/Main_Page. Acesso em: set. 2016

Laboratorio de Garagem, (2014), Tutorial: Como utilizar o Ethernet Shield com Arduino, Disponível em: http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-ethernet-shield-com-arduino, Acesso em: set. 2016

Miao, W., et. al. (2010) “Research on the architecture of Internet of things”. In.: 3rd IEEE International Conference on Advanced Computer Theory and Engineering (ICACTE). Sichuan, China, 21 ago. 2010. p. 484-487.

Pereira, C. R. (2014). “Aplicações web real-time com Node.Js”. Editora Casa do Código.

Node.js (2016) “Node. Js”. Disponível em: <https://nodejs.org/en/>. Acesso em: ago 2016.

PedroHS (2015) “Controlando Leds via página web utilizando Node.js e Arduino”. Disponível em: <https://pedrohs.github.io/controle-de-leds-via-http/>. Acesso em: set. 2016

Perez, A. L. F. et al. (2013) “Uso da Plataforma Arduino para o Ensino e o Aprendizado de Robótica”. Proceedings of the International Conference on Interactive Computer aided Blended Learning (ICBL2013).

Rea, S.; Aslam, M. S.; Pesch, D. (2013) “Serviceware-A service based management approach for WSN cloud infrastructures”. In: IEEE INT. Conf. Pervasive Comput. Commun. Work. Percom Work. 2013, 2013. Anais. . . [S.l.: s.n.], n. March, p.133–138.

Ribeiro, M. I. (2004) “Sensores em Robótica”, Enciclopédia Nova Activa Multimédia, Volume de Tecnologias, pags. 228-229. Portugal.

Robot@escola. (2016) “Tutorial de Programação Arduino – Lição 3”, Disponível em: http://escoladerobotica.ipcb.pt/?page_id=297. Acesso em: set. 2016

Souza, F., (2013) “Arduino - Primeiros Passos”, Disponível em: http://www.embarcados. com.br/arduino- primeiros-passos, Acesso em: Set. 2016.

Vermesan O. e Friess P., (2013) “Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems “