Aplicação em Tempo Real de Monitoramento de Umidade e Temperatura Utilizando *Arduino*

Rosiene Oliveira Dilly, Luiz Felipe Carvalho Mendes

Sistemas de Informação - Centro de Ensino Superior de Juiz de Fora
Juiz de Fora - MG - Brasil

rosiene.dilly@gmail.com, luizfelipe.carvalho.mendes@gmail.com

Abstract. *For a company that offers document management services, the paper still has great importance for businesses that choose to hire this type of service. But at the same time, the paper is very susceptible to natural, chemical and biological agents causing its life is decreased and deterioration in some cases reaches the total loss of information may cause immeasurable damage to businesses because of inspections and actions court. So for companies that provide custody services of these documents, the constant monitoring of two agents is important: Temperature and humidity. The high cost of the equipment, because it takes several units over the sheds, proprietary software and difficulty of integration with existing systems led to the development of this pilot using the open source Arduino hardware. This paper presents the development of an application that collects humidity and temperature measurement data through the Arduino and its components, recording in the database and in providing Web interface, the information in real time.*

Resumo. *Para uma empresa que oferece serviços de gestão documental, o papel ainda possui grande importância para as empresas que escolhem contratar este tipo de serviço. Mas, ao mesmo tempo, o papel é muito suscetível a agentes naturais, químicos e biológicos fazendo com que sua vida útil seja diminuída e sua deterioração em alguns casos alcança a total perda de informação podendo causar prejuízos imensuráveis aos negócios por causa de fiscalizações e ações judiciais. Portanto, para empresas que prestam o serviço de custódia destes documentos, é importante o monitoramento constante de dois agentes: temperatura e umidade. O alto custo dos aparelhos, pois são necessários várias unidades ao longo dos galpões, softwares proprietários e dificuldade de integração com sistemas já existentes motivou o desenvolvimento deste piloto utilizando o hardware open source Arduino. Este trabalho apresenta o desenvolvimento de uma aplicação que coleta dados de medição de umidade e temperatura através do Arduino e seus componentes, gravando em banco de dados e disponibilizando em interface Web as informações em tempo real para os gestores da empresa.*

1. Introdução

Empresas especializadas em gestão de documentos, por recomendação de órgãos regulamentadores, devem manter o ambiente no qual os documentos são armazenados, sob controle e monitoramento constante. Esse controle e monitoramento se dá através de circuito interno de TV, acesso restrito ao local de custódia dos documentos e também

de agentes que podem acelerar a deterioração do papel. O papel sofre ação de diversos agentes que contribuem para a diminuição da sua vida útil e em casos extremos podem levar a perda de informação, muitas vezes cruciais às instituições geradores destes documentos causando prejuízos financeiros enormes. A umidade e temperatura são exemplos destes agentes que devem ser monitorados, CONARQ (2014).

Os dispositivos existentes no mercado para realização de medição de temperatura e umidade são de custo elevado e extremamente dependente da variação cambial, pois normalmente, poucos componentes são produzidos no Brasil. Além disto, são pouco flexíveis, pois estão interligados a softwares e protocolos proprietários. Posto isto, e devido ao baixo custo do hardware *Arduino*, a facilidade e flexibilidade de utilização e programação, permitindo o funcionamento em módulos para adicionar novos recursos, foi desenvolvido um projeto utilizando módulo de sensor de umidade e temperatura conectado a uma placa *Arduino*.

O *Arduino*, desde o seu inicio em 2005, vem se destacando por ser uma ferramenta de hardware livre, *Arduino* (2015). Se destaca também pela facilidade para desenvolver um projeto, pois possui uma documentação vasta e uma comunidade bastante ativa pela Internet. Desta forma, mesmo pessoas que não possuam conhecimento profundo em eletrônica e programação conseguem desenvolver projetos que utiliza placas *Arduino* e módulos compatíveis.

A solução proposta para o problema em questão além de utilizar o *Arduino* para realizar o controle das medições de umidade e temperatura oferece uma aplicação Web capaz de receber essas leituras, armazenar e exibir as informações em tempo real e históricas.

2. *Arduino*

Arduino é uma plataforma de computação física ou embarcada de código aberto baseado em hardware e software de fácil utilização, onde pode ser programada para processar as entradas e saídas entre o dispositivo e os componentes externos conectados a ele, Banzi (2012). O *Arduino* foi pioneiro ao lançar um hardware totalmente livre, o que atraiu um número muito grande de pessoas interessadas em desenvolver e contribuir com o mesmo.

O *Arduino* é utilizado na construção de diversos projetos de automação, robótica, entre outros. Devido principalmente a facilidade de sua utilização, bem como também as comunidades ativas e sempre disposta a compartilhar projetos, códigos e artigos na internet que auxiliam muito na construção de um novo projeto.

Como exemplo de projetos desenvolvidos em *Arduino* temos um *Carro Arduino Controlado pela Internet*, que com a utilização de alguns componentes e programação foi desenvolvido esse projeto, Carlos Romeral (2013). E outro exemplo é o *Protetor de Gatos*, para evitar que gatos se aproximem de uma determinada área, e através de sensor de movimento, quando o gato se aproximar é emitido um áudio exatamente igual ao do dono de reprovação, Luckyresistor (2014).

O surgimento do *Arduino* partiu da procura de um meio barato e a fim de facilitar a inserção de estudantes de design em tecnologia no *Interaction Design Institute* na cidade de Ivrea, Itália. O professor Massimo Banzi, em 2005, junto com David Cuartielles, um pesquisador visitante da Universidade de Malmõ, na Suécia, que

também estava a procura de uma solução parecida, foram os que deram os primeiros passos ao criar algo que foi batizado de *Arduino*. Os produtos existentes, na época, eram caros e relativamente difíceis de usar. Banzi e Cuartielles então, decidiram desenvolver um microcontrolador que poderia ser utilizado pelos seus estudantes de arte e design em seus projetos. Os principais fatores eram que fosse uma plataforma que qualquer pessoa pudesse utilizar e que fosse barato. David Cuartielles desenhou uma placa, e um aluno de Massimo, David Mellis, programou o software para executar a placa. Massimo contratou um engenheiro local, Gianluca Martino, para produzir uma tiragem inicial de duzentas placas, Evans, Noble e Hochenbaum (2013).

A placa foi chamada de *Arduino* em referencia a um bar local frequentado por membros do corpo docente e alunos do instituto, o bar é chamado *"Bar Di Re Arduino"* em homenagem ao Rei *Arduin* que governou a Itália no ano de 1002 d.C.. As placas eram vendidas em forma de kit para que os alunos fizessem seus próprios projetos. A tiragem inicial foi rapidamente vendida, e mais unidades foram produzidas para manter a demanda. A popularidade do *Arduino* cresceu rapidamente quando o grande público percebeu que o hardware era um sistema de fácil utilização, de baixo custo e que poderia ser usado em seus projetos, bem como era uma excelente introdução a programação de microcontroladores, Evans, Noble e Hochenbaum (2013). O projeto original foi melhorado e novas versões foram introduzidas.

2.1. Hardware

Ao longo dos anos, várias versões de placas foram lançadas e o modelo utilizado no projeto é o modelo *Uno*. Na tabela 1 estão todos os modelos de *Arduino*, e suas especificações, existentes até o momento:

Tabela 1. Versões do *Arduino*

Name	Processor	Operating Voltage/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	USB
Uno	ATmega328	5 V / 7-12 V	16MHz	6/0	14/06	1	2	32	Regular
Due	AT91SAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	-	96	512	2 Micro
Leonardo	ATmega32u4	5 V / 7-12 V	16MHz	12/0	20/7	1	2.5	32	Micro
Mega 2560	ATmega2560	5 V / 7-12 V	16MHz	16/0	54/15	4	8	256	Regular
Mega ADK	ATmega2560	5 V / 7-12 V	16MHz	16/0	54/15	4	8	256	Regular
Micro	ATmega32u4	5 V/7-12 V	16MHz	12/0	20/7	1	2.5	32	Micro
Mini	ATmega328	5 V / 7-9 V	16MHz	8/0	14/6	1	2	32	-
Nano	ATmega168	5 V / 7-9 V	16MHz	8/0	14/6	0.512	1	16	
	ATmega328					1	2	32	Mini-B
Ethernet	ATmega328	5 V / 7-12 V	16MHz	6/0	14/4	1	2	32	Regular
Esplora	ATmega32u4	5 V / 7-12 V	16MHz	-	-	1	2.5	32	Micro
ArduinoBT	ATmega328	5 V / 2.5-12 V	16MHz	6/0	14/6	1	2	32	-
Fio	ATmega328P	3.3 V / 3.7-7 V	8MHz	8/0	14/6	1	2	32	Mini
Pro (168)	ATmega168	3.3 V / 3.35-12 V	8MHz	6/0	14/6	0.512	1	16	-

Pro (328)	ATmega328	5 V / 5-12 V	16MHz	6/0	14/6	1	2	32	-
Pro Mini	ATmega328	3.3 V / 3.35-12 V	8MHz	6/0	14/6	0.512	1	16	-
		5 V / 5-12 V	16MHz						
LilyPad	ATmega168V	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-
	ATmega328V								
LilyPad USB	ATmega32u4	3.3 V / 3.8-5V	8MHz	4/0	9/4	1	2.5	32	Micro
LilyPad Simple	ATmega328	2.7-5.5 V / 2.7-5.5 V	8MHz	4/0	9/4	1	2	32	-
LilyPad SimpleSnap	ATmega328	2.7-5.5 V / 2.7-5.5 V	8MHz	4/0	9/4	1	2	32	-
Yun	ATmega32u4	5 V	16MHz	12/0	20/7	1	2.5	32	Micro

O maior diferencial entre o *Arduino Uno*, figura 1, e seus antecessores é a inclusão de um micro controlador programado *ATmega328* como um conversor USB-para-serial o que significa que o Uno pode ser reprogramado para se parecer com outro dispositivo USB, tal como mouse, teclado ou *joystick*.

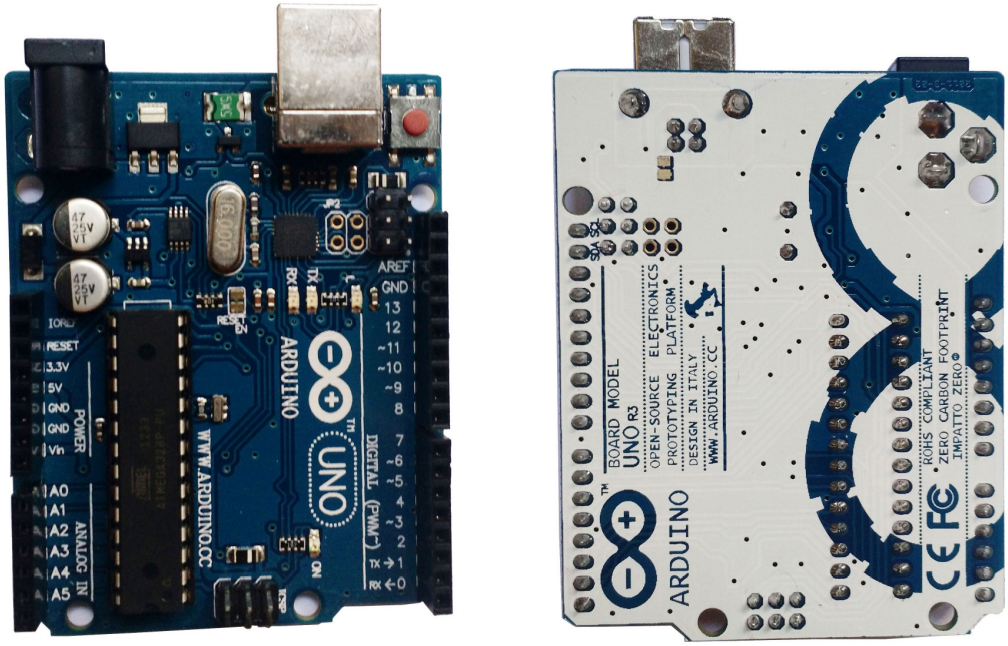

Figura 1. Placa *Arduino Uno*

A placa *Arduino Uno* possui diversos conectores para plugar componentes externos. Vejamos como estão organizados os pinos na placa:

- 14 pinos de entrada e saída digital (0-13): Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto e conforme foi definido no *sketch* criado na IDE.
- 6 pinos de entradas analógicas (A0 - A5): Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V onde serão convertidos para valores entre 0 e 1023.
- 6 pinos de saídas analógicas (3, 5, 6, 9, 10 e 11): São pinos digitais que podem ser programados para ser utilizados como saídas analógicas, utilizando modulação PWM.

A alimentação da placa Uno pode ser feita a partir da porta USB do computador ou através de um adaptador AC-CC, ou seja, um adaptador de Corrente Alternada para Corrente Contínua. Para o adaptador a tensão deve ser de 9 a 12 volts.

2.2. Linguagem e Ambiente de Programação

A linguagem de programação para o *Arduino* é baseada nas linguagens C/C++ E seus programas são chamados de *sketch*, ou seja, o arquivo que contem as linhas de códigos de instrução que são compiladas, enviadas e executadas em uma placa *Arduino*.

A integração de comunicação do *Arduino* com qualquer outra aplicação de linguagem diferente se faz através dos componentes de comunicação que o *Arduino* dispõe, no trabalho proposto o *Arduino* possui o componente *Ethernet Shield*, figura 2, o que possibilita a comunicação com qualquer aplicação via rede.

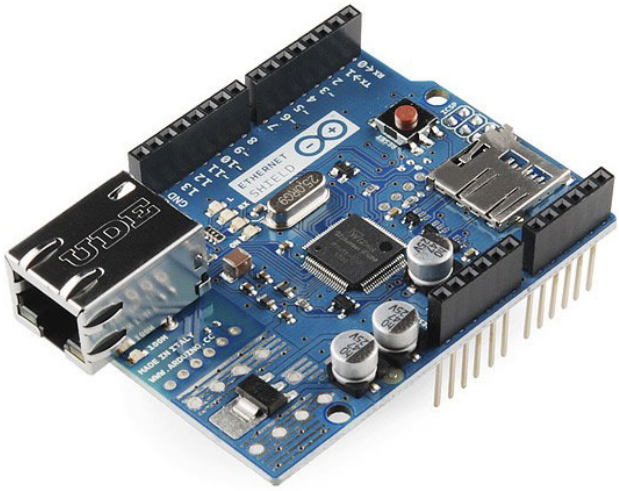

Figura 2. Módulo *Ethernet Shield*

Um *sketch* típico consiste de duas partes ou rotinas: a primeira é a rotina de inicialização chamada *setup*, e a segunda é a rotina chamada *loop*, que geralmente contém o corpo principal do código.

O *Arduino* deve ser preparado ou configurado antes que você possa trabalhar com ele. Essas configurações são colocadas dentro de uma rotina de inicialização ou função adequadamente chamada *setup*. Coisas típicas que você deve fazer no *setup*

incluem a inicialização dos pinos digitais e definição da taxa de transmissão para a comunicação serial.

A função *setup* deve sempre existir, mesmo quando não for utilizada, ou um erro será gerado ao verificar ou carregar um *sketch*.

Já a segunda função pré-definida de um *sketch* do *Arduino* é a função chamada *loop*, que é executada logo após a função *setup* e a função é invocada continuamente enquanto o *Arduino* estiver alimentado de energia elétrica.

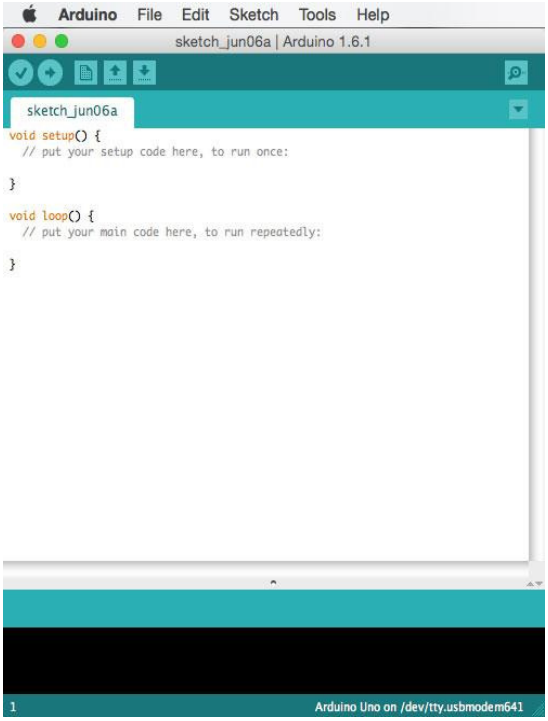

Figura 3. IDE *Arduino*

O *open source Arduino Software* (IDE) é o responsável por compilar o código e enviar o mesmo para o micro controlador. O ambiente de desenvolvimento é compatível com os principais sistemas operacionais: Windows, Mac OS X e Linux. O ambiente é escrito em Java e baseadas em *Processing*, Evans, Noble e Hochenbaum (2013). O software pode ser usado pra qualquer placa *Arduino*.

Em um desenvolvimento com *Arduino* normalmente são utilizados outros componentes eletrônicos, alguns simples como LED's e outros mais complexos como placas de rede, sensores, motores, entre outros componentes eletrônicos. No caso dos componentes mais complexos, é normal existir bibliotecas especificas para os componentes de forma a facilitar o desenvolvimento da mesma.

O sensor de qualidade do ar, Figura 4, é projetado para testar a qualidade do ar e gases presentes no ambiente.

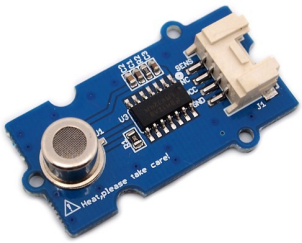

Figura 4. Módulo Sensor de Qualidade do Ar

Outro componente para o *Arduino* muito utilizado são os módulos de *displayers* LCD. Na figura 5, o modelo LCD é um *Shield*, ou seja, é acoplado em cima da placa do *Arduino*.

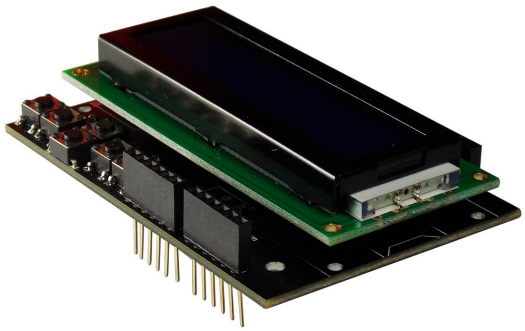

Figura 5. Módulo LCD *Shield* I2C

Para automação residencial o componente rele é muito utilizado, pois ele funciona como uma chave que é acionada na ausência ou presença de luz no ambiente. A Figura 6, é um modelo de rele para o *Arduino*.

Figura 6. Módulo Rele

Outro componente para automação residencial é o módulo de sensor de movimento e presença, figura 7, combinado com uma sirene, funcionando como um alarme na presença ou movimento de alguém quando acionado.

Figura 7. Módulo Sensor de Presença e Movimento

Com módulo de GPS para o *Arduino* torna fácil a localização de sua posição com precisão de poucos metros. Este componente pode ser muito utilizado em automóveis, no caso de furtos. Na figura 8, o módulo GPS é um *shield* e está acoplada ao *Arduino*, mas existem diferentes formatos de módulos com a mesma função.

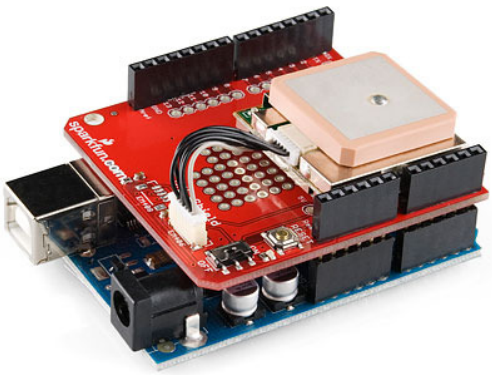

Figura 8. Módulo GPS *Shield*

Como um sensor de estacionamento, o módulo sensor de distância por ultrassom, figura 9, é muito eficiente com uma ótima precisão de 2 centímetros até 4 metros.

Figura 9. Módulo Sensor de Distância por Ultrassom

Outro módulo muito conhecido do *Arduino* é o módulo *Bluetooth*, figura 10, que se conecta com outros dispositivos com tecnologia *Bluetooth* e quando conectados realizam transmissões de dados entre eles.

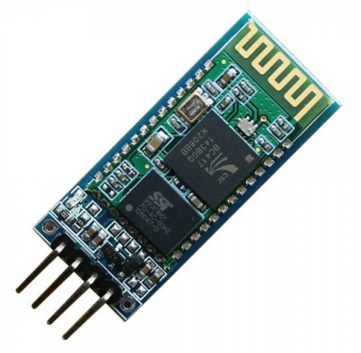

Figura 10. Módulo Bluetooth

São inúmeros os módulos para o *Arduino*, oferecendo diversas possibilidades de desenvolvimento de projetos eletrônicos integrado com seus módulos.

3. Estudo de Caso

A Célula Gestão de Documentos é uma empresa especializada em gestão documental, atuando no mercado desde 2004, com uma carteira de mais de 180 clientes de todos os portes localizados nos estados de Minas Gerais, Rio de Janeiro, Sergipe, Mato Grosso do Sul e São Paulo. Quando se fala em gestão de documentos, serviços como digitalização e guarda de arquivos são os mais comuns. No caso da guarda especificamente, a Célula tem uma responsabilidade enorme sobre os documentos que guarda de seus clientes e daí a importância em monitorar de forma constante todo e qualquer agente que de alguma forma possa estar em contato com o papel.

Atualmente, a Célula possui 2 galpões, de aproximadamente 750 metros quadrados, com capacidade para 300.000 caixas do tipo arquivo. O pé-direito com altura de 13 metros e os vãos livres auxiliam em um controle natural de temperatura e umidade, porém é sabido que as caixas que estejam mais próximas do telhado é natural que estejam em uma temperatura um pouco maior das que se encontram nos pontos mais baixos. Cada galpão possui 8 módulos de estantes em metal com um corredor entre elas de aproximadamente 1,50 metro para circulação do elevador eletro-hidráulico.

Com esta estrutura, é importante portanto colocar ao menos 15 sensores em cada estante para ter um levantamento detalhado de como a temperatura varia dentro de cada galpão e entre cada estante. Levando em consideração o custo de um leitor de temperatura simples, em torno de R$ 200,00 cujo os dados precisam ser descarregados via USB toda vez que deseja-se montar um relatório com as últimas leituras. Caso deseje saber em tempo real, é preciso que se vá em cada sensor e visualmente tenha esta informação. Quando falamos de sensores automáticos, que possam enviar estas informações, ou que gravam na rede, os preços ficam ainda mais inviáveis e pela quantidade de sensores inviabiliza qualquer monitoramento de temperatura e umidade.

Mas por que este monitoramento é tão importante? Os documentos gerados pelas empresas, sejam privadas ou públicas, possuem uma vida útil, que na terminologia adequada, chama-se temporalidade, CONARQ (2014). Esta temporalidade define por quanto tempo a empresa precisa guardar o documento em papel, seja a pasta de um funcionário, uma nota fiscal emitida ou recebida, e após atingir este tempo, qual será a destinação final deste documento: guarda permanente ou possuem um prazo para

expurgo podendo variar de acordo com o tipo de documento. Segundo a Resolução 14, do Arquivo Nacional, a título de exemplo, pastas de servidores federais devem ser guardadas por 100 anos a partir da sua admissão, e só após esta data, podem então, ser eliminados. Por outro lado, plantas e desenhos técnicos de engenharia e arquitetura, por exemplo, são documentos de guarda permanente. Sendo assim, comprova-se a importância por parte da empresa que detém a custódia destes documentos, em controlar de perto o ambiente onde este documento está armazenado para que o papel não se deteriore seja por agentes naturais, químicos ou biológicos. Para tal, o ambiente deve ser constantemente dedetizado e o controle e monitoramento da temperatura e umidade devem ser constantes para que medidas sejam tomadas caso os níveis adequados não sejam mantidos.

Nos depósitos de guarda permanente de documentos textuais (papel), são recomendados índices de umidade relativa (UR) de 50%, com variação diária de +/- 5%. A temperatura ideal para os documentos é de 20º C, com variação diária de +/- 1ºC, obtida por meio do uso individual, ou combinada de ventilação natural ou forçada, de desumidificadores ou de unidades de refrigeração, CONARQ (2014).

Existem diversos dispositivos no mercado para realização de medições de umidade e temperatura, porém o custo desses dispositivos é alto e o objetivo do trabalho é mostrar que é possível construir um dispositivo mais barato e flexível.

Um dos sensores de umidade e temperatura utilizado pela empresa Célula Gestão de Documentos é o modelo RHT10 da empresa Extech, e está localizado no Data Center. Ele é configurado previamente para definir os intervalos de medição, que pode ser de minuto em minuto, hora em hora e dia a dia. Quanto menor o intervalo mais rápido sua memória interna alcança sua capacidade máxima. O problema é que para realizar a gravação dos dados e visualizar as últimas leituras é preciso ir até o local, retirar o sensor da parede e plugar o mesmo a uma porta USB de um computador que já deve estar com o programa disponibilizado pelo fabricante instalado para salvar os dados em um arquivo e gerar os relatórios de medições. Este processo de gravação dos dados é feito semanalmente, pois o sensor está configurado para realizar medições a cada 1 minuto. O preço atual do modelo RHT10 é de R$ 631,80 (seiscentos e trinta e um reais e oitenta centavos).

Figura 11. Extech RHT10

Além do RHT10, a Célula também utiliza o sensor modelo ITLOG-80, ele possui um visor, onde é possível checar as informações de umidade e temperatura atual. A Célula, em julho de 2014, adquiriu 7 sensores desse modelo que estão colocados nas estantes do primeiro galpão. E assim como o sensor modelo RHT10, as informações de

temperatura e umidade também precisam ser coletadas através da USB através de software proprietário. O preço atual deste modelo é R$ 291,30 (duzentos e noventa e um e trinta), consultado no site instrutemp.com.br (2015).

Figura 12. ITLOG-80

Outro dispositivo com a mesma função é o Incoterm 3030.22, muito semelhante ao ITLOG-80. Ele possui o visor com a umidade e temperatura atual e os dados históricos das medições são gravados em memória e é preciso descarregar esses dados através da USB de um computador.

Figura 13. Incoterm 3030.22

O custo do modelo Incoterm 3030.22 é de R$ 2.212,99 (dois mil, duzentos e doze reais e noventa e nove centavos), Ponto Frio (2015).

Desta forma o *Arduino*, por ser um hardware de código aberto, e contar com diversas bibliotecas também de código aberto, é uma ferramenta flexível e eficiente na solução do problema proposto e com um baixo custo. O custo total do Arduino, bem

como seus componentes utilizados para a construção do projeto foi de apenas R$ 50,48 (cinquenta reais e quarenta e oito centavos).

4. Desenvolvimento da Aplicação

O projeto foi desenvolvido em duas partes: primeiramente a montagem dos componentes e em seguida o desenvolvimento da aplicação responsável em armazenar as leituras e exibir as informações históricas e em tempo real. Ambos serão detalhados nos tópicos a seguir:

4.1. Projeto de Hardware

Para o projeto de hardware foi utilizada uma placa de rede *Ethernet Shield* para enviar os dados coletados na medição de temperatura e umidade para o servidor.

Placas *Shield*, figura 14, são placas de expansão de hardware que encaixam na placa *Arduino* principal.

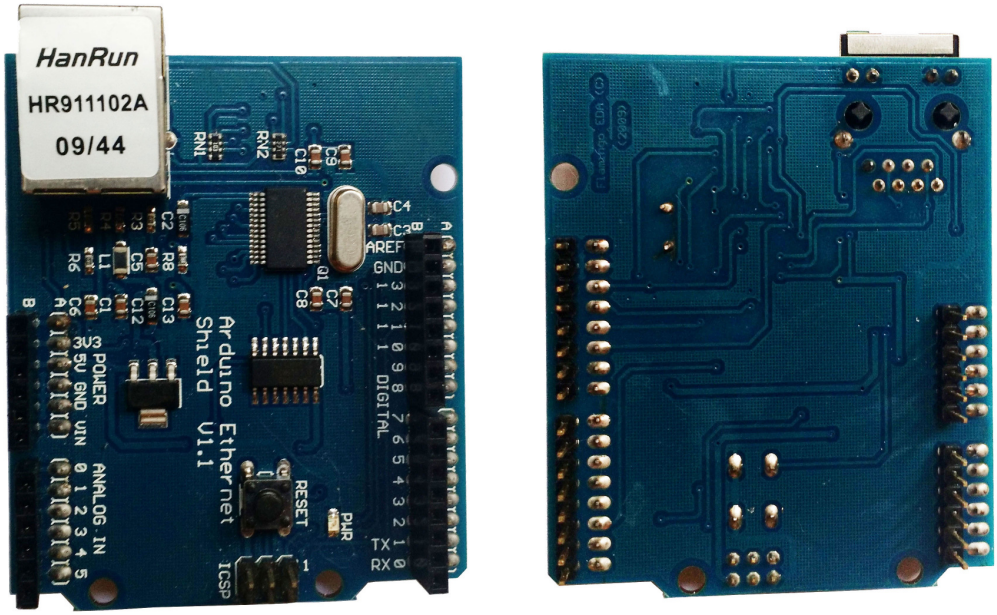

Figura 14. Placa *Ethernet Shield*

O sensor de umidade e temperatura modelo *DHT11*, figura 15, é um pequeno dispositivo que possui quadro pontos de conexão: um para receber energia, outro para saída, um nulo e o para transmitir as informações de medição.

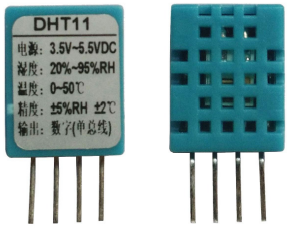

Figura 15. Sensor DHT11

Além dos módulos *Ethernet Shield* e o sensor de umidade e temperatura *DHT11*, foi utilizada a *protoboard*, ou placa de ensaio que é uma placa com furos e conexões condutoras para montagem de circuitos elétricos experimentais. Também foi utilizado um resistor responsável por diminuir a corrente que chega ao sensor e 3 cabos para fazer as conexões.

Na figura 16 é exibido o modelo de conexões da placa e seus componentes. A placa *Ethernet* fica acoplada a placa do *Arduino*. Conexões com o módulo sensor *DHT11*:

- Cabo vermelho: está com uma ponta conectada no pino *5V* do *Arduino* e a outra ponta na linha do resistor e da ponta de alimentação do sensor;
- Cabo preto: é o neutro e está com uma ponta conectada no *GND* e a outra conectada na ponta do neutro do sensor;
- Cabo amarelo: é a conexão que vai transmitir os valores de retorno do sensor, e uma das pontas está no pino digital 7 do *Arduino*.

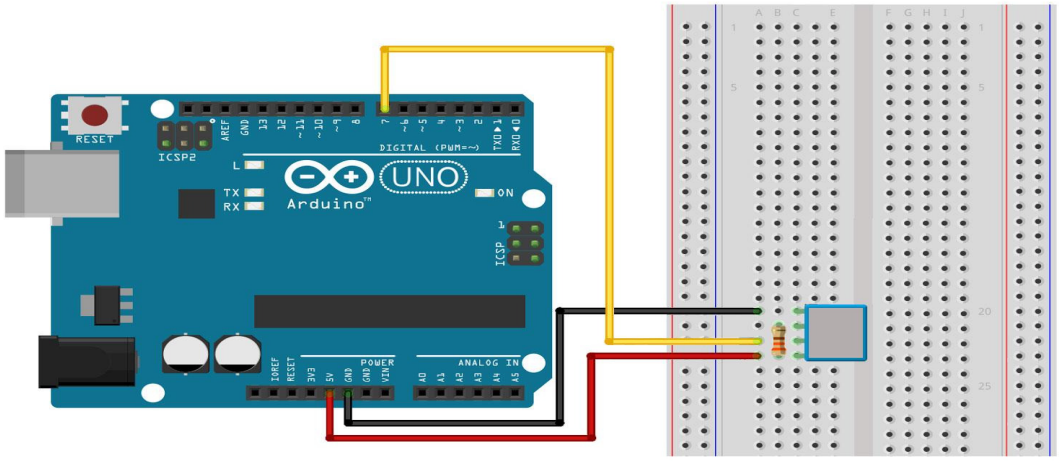

Figura 16. Modelagem do Projeto

O circuito eletrônico completo conforme o modelo, onde pode ser visto na figura 17. O *Arduino* está conectado através da *protoboard* e cabo aos seus componentes.

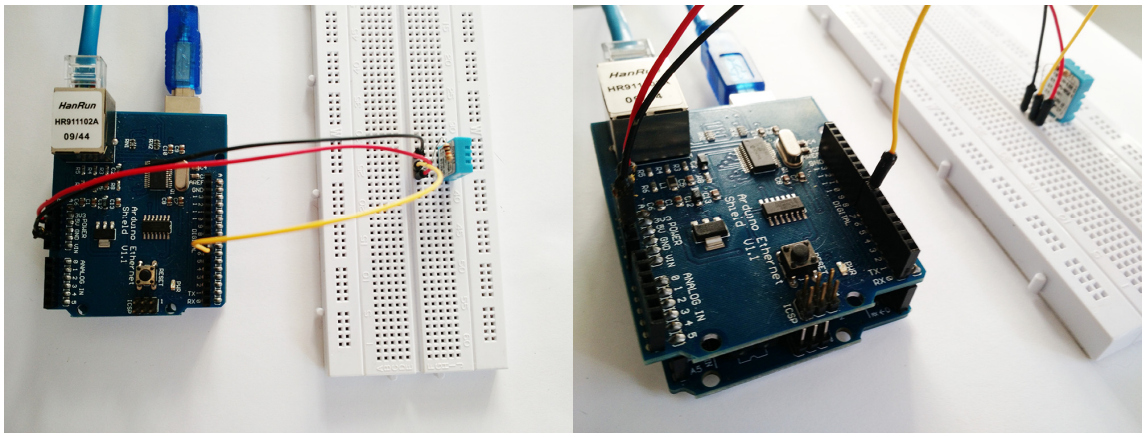

Figura 17. Circuito Eletrônico

4.2. Desenvolvimento das Aplicações

A primeira aplicação foi desenvolvida para o *Arduino*, para executar basicamente duas funções: a função de coleta das informações do sensor de umidade e temperatura DHT11 e o envio dessas informações através do componente *Ethernet Shield*, que conecta no servidor e envia as informações via *GET*.

Na figura 18, é apresentado o exemplo de como seria apenas a coleta das informações do sensor de umidade e temperatura DHT11, é possível observar que devido ao include da biblioteca do componente dht11.h, torna-se simples a utilização do componente. Na primeira linha de comando do loop é feita a leitura dos dados do sensor no qual é informado qual pino digital o sensor está conectado, e no nosso exemplo ele está conectado no pino 7.

```
// importando biblioteca de umidade e temperatura
#include <dht11.h>

// variavel que representa o sensor
dht11 sensor;

//setup de inicializacao
void setup() {
  Serial.begin(9600);
}

void loop() {

  //variavel chk recebe um verificador
  int chk = sensor.read(7);

  //verificacao de erro de leitura do sensor
  switch(chk) {
    case DHTLIB_OK:
      Serial.println("OK");
      break;
    case DHTLIB_ERROR_CHECKSUM:
      Serial.println("Erro no checksum");
      break;
    case DHTLIB_ERROR_TIMEOUT:
      Serial.println("Tempo esgotado");
      break;
    default:
      Serial.println("Erro desconhecido");
  }

  //imprimindo dados de umidade e temperatura
  Serial.println(sensor.humidity);
  Serial.println(sensor.temperature);

  delay(5000);
}
```

Figura 18. Código de Leitura do Módulo Sensor DHT11

Após a leitura é feita uma série de checagens no qual é verificado se os dados retornados pelo sensor estão corretos. Por fim é exibido no Serial Monitor do *Arduino* as informações de umidade e temperatura. E o processo se repete enquanto o *Arduino* estiver sendo alimentado de energia.

Para enviar as informações na rede através do componente *Ethernet Shield*, também utiliza biblioteca para realizar a conexão com o servidor, e neste caso é preciso

primeiramente configurar no *setup* do *sketch* a inicialização do *Ethernet Shield* passando o endereço MAC, já definido, como parâmetro, na figura 19 o exemplo da conexão.

```
// importando biblioteca de Ethernet
#include <UIPEthernet.h>

// variavel que represente o cliente da Ethernet
EthernetClient client;

// byte[] do mac address do dispositivo
byte mac[] = { 0x54, 0x34, 0x41, 0x30, 0x30, 0x31 };

void setup() {
  // configuracao do shield de ethernet para se comunicar
  Ethernet.begin(mac);
}

void loop(){
  //ip do servidor
  char server[] = "192.168.0.110";

  if (client.connected()) { client.stop(); }

  //conectando com o servidor
  if (client.connect(server, 3000)) {
    //Enviando requisicao para o servidor
    client.print("GET /reading/new?");
    client.print("temperature=");
    client.print(sensor.temperature);
    client.print("&&");
    ...
    client.stop();
  }
}
```

Figura 19. Exemplo de Código de Conexão com Servidor

Na função *loop* é passado o endereço do servidor no qual o *Arduino* vai se conectar e em seguida é feita a conexão, e se a conexão for bem sucedida são passadas as informações através do *GET*.

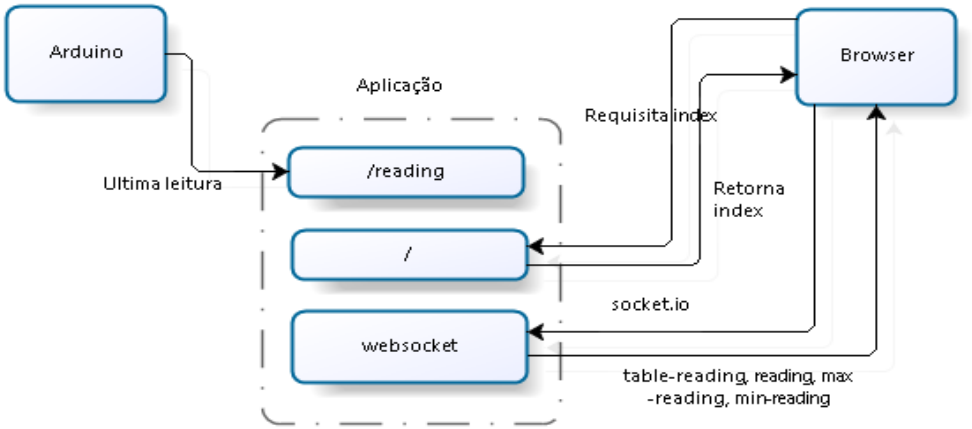

Figura 20. Arquitetura da Solução

Para receber, gravar e exibir as informações de medição de umidade e temperatura foi desenvolvida uma aplicação em *JavaScript* utilizando a plataforma *Node.js* com o *Framework Express.js*.

O *Node.js* é uma plataforma construída em tempo de execução em *JavaScript* do *Chrome* para facilitar a construção rápida de aplicações de rede. O *Node.js* é orientado a eventos que o torna leve e eficiente, perfeito para aplicações em tempo real de dados intensivos que são executados através de dispositivos distribuídos, *Nodejs* (2015).

E o *Express* é um Framework de aplicações Web minimalista e flexíveis para o *Node.js* que fornece um conjunto robusto de recursos para Web e aplicações móveis, *Expressjs* (2015).

A comunicação cliente-servidor é feita através de *sockets*, que proporcionam um canal de comunicação bidirecional entre o servidor e os clientes. Sempre que o servidor receber uma mensagem, ele vai enviá-la para todos os outros clientes conectados. *Socket.io* é uma ferramenta construída sobre o *Node.js* que permite essa comunicação cliente-servidor, *Socket.io* (2015).

As medições coletadas através do sensor são enviada do *Arduino* para a aplicação, e na aplicação são gravadas em banco de dados *Sqlite*. O *Sqlite* é uma biblioteca que implementa um banco de dados relacional. A biblioteca lê e escreve diretamente em um arquivo de banco de dados. E uma ferramenta livre para uso independente, comercial ou privado. Atualmente é o mais amplamente implementado banco de dados do mundo, *Sqlite* (2015).

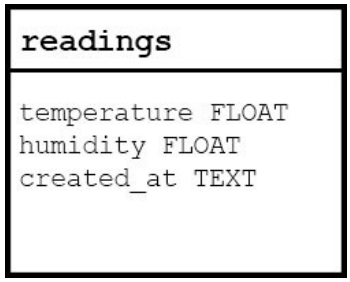

Figura 21. Tabela *Readings*

O *Arduino* realiza a medição de umidade e temperatura e em seguida faz uma requisição *http* para o servidor Web, através do *Socket.io*, o servidor notifica todos os clientes (navegador) desta nova leitura. E desta forma, a informação é exibida em tempo real no navegador conforme figura 22.

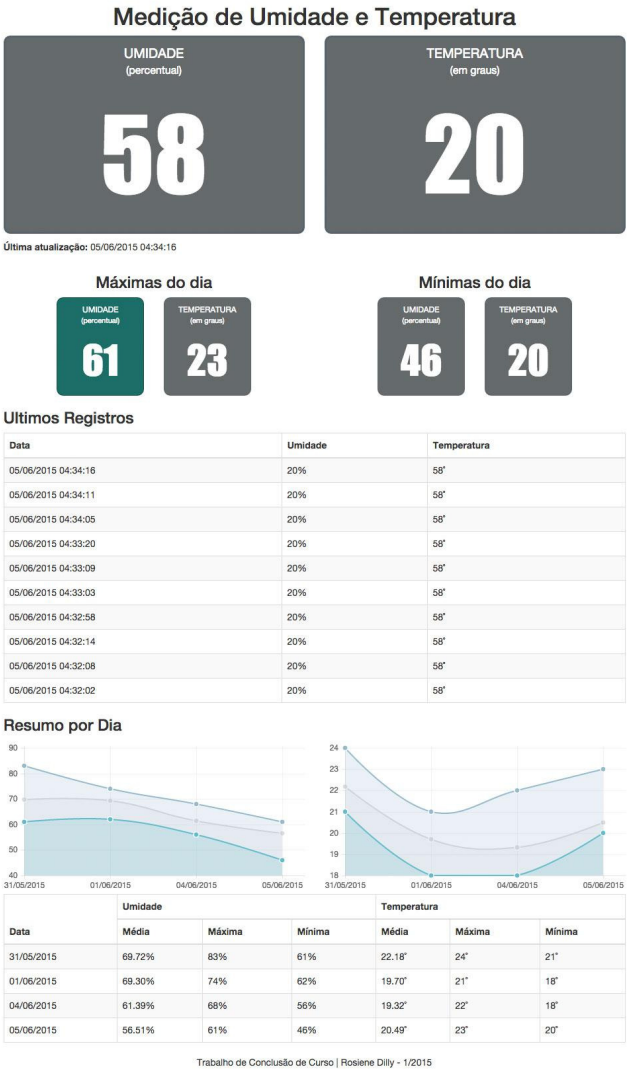

Figura 22. Interface da Aplicação

Na interface da aplicação, figura 22, são exibidas a as ultimas coletas de medições de umidade e temperatura atualizadas em tempo real (*real time*). Como também são exibidas as máximas e mínimas de umidade e temperatura do dia. E os gráficos e a ultima tabela, na interface da aplicação, trazem um resumo agrupado por dia da média, máxima e mínima de umidade e temperatura.

5. Considerações Finais

Devido à necessidade da empresa Célula em realizar o monitoramento de umidade e temperatura no espaço onde os documentos são armazenados, e visto que, os dispositivos disponíveis têm um custo elevado e são pouco flexíveis, viu-se que o desenvolvimento de uma solução baseada em hardware barato e *open source* pode ser uma saída de bom custo benefício.

A posição oficial da empresa é que o projeto possui um potencial bastante interessante para ser adotado no dia-a-dia. A questão do baixo custo aliado a possibilidade de um software ser desenvolvido de forma independente e a possível modularização do *Arduino* que permite a inserção de novos sensores é o maior atrativo.

Ao mesmo tempo, para ser adotado, a empresa acredita que dois problemas devem ser resolvidos: o fornecimento da energia que hoje é realizado através da USB deve ser substituído pelas baterias padrão dos outros medidores e que a leitura das medições possam ser transmitidas via rede sem fio ou Bluetooth, pois a instalação de pontos de rede e de energia nas estantes onde os documentos são guardados são inviáveis, tanto do ponto de vista de segurança (pontos de energia próximos a papel) quanto de infraestrutura (o galpão não possui nenhum tipo de cabeamento justamente para evitar qualquer possível curto). As duas questões são relativamente simples de se resolver, pois a transmissão via Bluetooth ou rede sem fio remete a adição de um novo módulo, o que ainda não encareceria o projeto a ponto de não justificar a mudança e a bateria também tem o custo extremamente acessível mas exigirá um conhecimento de eletrônica um pouco mais avançado.

O *Arduino* mostrou ser uma solução de custo acessível e que em comunicação com a aplicação em tempo real pôde-se construir uma solução flexível e permitindo aos gestores fácil acesso aos dados sem a dependência e custo de um software proprietário. Todos os componentes foram adquiridos fora do Brasil, sem incidência de impostos.

A tabela 2 mostra o custo dos componentes adquiridos para o *Arduino* contra os medidores já prontos encontrados no mercado. Enquanto o valor do *Arduino*, somado com os componentes é de R$ 50,48 (cinquenta reais e quarenta e oito centavos), o *datalogger* ITLOG-80 custa mais que o dobro, enquanto o RHT10 custa mais de dez vezes o valor do projeto. Mesmo utilizando uma alíquota de 100% de impostos sobre o valor dos componentes chegaríamos a um valor total de R$ 100,96 (cem reais e noventa e seis centavos), ainda inferior ao *datalogger* ITLOG-80. Além disto, o *Arduino* permitiu a construção de uma aplicação nos moldes desejados pela empresa e já está pronto para a inclusão de outros módulos.

Tabela 2. Comparativo de Preços *Arduino* X Dataloggers

Componente	Valor em Dólar	Valor em Reais
Arduino Uno Ethernet Shield Sensor DHT11	16,18	50,48*
EXTECH RHT10		631,80
ITLOG-80		291,30
Incoterm 3030.22		2.212,99

*Cotação R$ 3,12 (15/06/2015). Fonte: Uol.

Nesse trabalho foi criada uma aplicação que coleta dados de medição de umidade e temperatura através do *Arduino*, grava as informações em banco de dados e disponibiliza as mesmas em tempo real via interface Web.

A principal dificuldade encontrada foi que inicialmente o *Arduino* estava configurado como servidor, e a aplicação requisitava ao *Arduino* as informações coletadas. Devida a limitação de *hardware*, essas requisições nem sempre eram atendidas e a solução encontrada, foi a inversão de papeis, ou seja, o *Arduino* é o

responsável por enviar as medições para o servidor, desta forma a sua fila de processos é melhor gerenciada.

Para trabalhos futuros é interessante a inclusão de novos módulos como os sensores de fumaça, de presença, de qualidade do ar e para alarmes sonoros quando eventos que saem do padrão ocorram os gestores possam trabalhar de forma ativa na resolução dos problemas.

Outro trabalho é a integração da aplicação com alguma ferramenta de monitoramento de rede como *Nagios*[1] ou *Zabbix*[2].

E por fim melhorar a usabilidade da interface da aplicação adicionando filtros, novos gráficos entre outras funcionalidades para melhor análise dos dados.

6. Referência

Arduino. Disponível em: http://arduino.cc. Acesso em Junho 2015.

Cat Protector | Lucky Resistor. Disponível em: http://luckyresistor.me/cat-protector/. Acesso em Junho 2015.

DATALOGGER PARA UMIDADE E TEMPERATURA CONECTOR USB | EXTECH RHT10. Disponível em: http://www.instrutemp.com.br/instrutemp/produto/238/data+logger+para+umidade+e+temperatura+conector+usb+extech+rht10. Acesso em Junho 2015.

Development Board w/ Data Cable for Arduino UNO R3 - Deep Blue (Cable-52cm) - Free Shipping – DealExtreme. Disponível em: http://www.dx.com/p/development-board-w-data-cable-for-arduino-uno-r3-deep-blue-cable-52cm-312887#.VX77vPlVhBc. Acesso em Junho 2015.

DHT11 Temperature / Relative Humidity Sensor Module for Arduino - Light Blue - Free Shipping - DealExtreme. Disponível em: http://www.dx.com/p/dht11-temperature-relative-humidity-sensor-module-for-arduino-light-blue-308651#.VX3yyRNVikp. Acesso em Junho 2015.

Dólar Comercial: Cotação de hoje, gráficos e tabelas - UOL Economia. Disponível em: http://economia.uol.com.br/cotacoes/cambio/dolar-comercial-estados-unidos/. Acesso em Junho 2015.

Ethernet Shield para Arduino (original). Disponível em: http://www.labdegaragem.org/loja/arduino-ethernet-shield.html. Acesso em Junho 2015.

Ethernet Shield V1.1 for Arduino (Works with Official Arduino Boards) - Free Shipping - DealExtreme. Disponível em: http://www.dx.com/p/ethernet-shield-v1-1-for-arduino-66908#.VX2bkBNViko. Acesso em Julho 2015.

Evans, M., Noble, J. e Hochenbaum J. (2013) “Arduino em Ação”, Em Novatec Editora.

[1] http://www.nagios.org
[2] http://www.zabbix.com

Expressjs. Disponível em: http://expressjs.com. Acesso em Junho 2015.

Extech RHT10 Humidity and Temperature USB Datalogger. Disponível em: http://www.calright.com/Products/prod_id/844/. Acesso em Junho 2015

Internet controlled Arduino car. Disponível em: http://purposefulscience.blogspot.com.es/2013/07/internet-controlled-arduino-car.html. Acesso em Junho 2015.

LCD Shield I2C. Disponível em: http://www.labdegaragem.org/loja/lcd-shield-i2c.html. Acesso em Junho 2015.

McRoberts M., (2011) "Arduino Básico", Em Novatec Editora.

Módulo Bluetooth - HC-06. Disponível em: http://www.labdegaragem.org/loja/modulo-bluetooth-data-link.html. Acesso em Junho 2015.

Módulo Rele. Disponível em: http://www.labdegaragem.org/loja/modulo-rele-10a.html. Acesso em Junho 2015.

Nodejs. Disponível em: http://nodejs.org. Acesso em Junho 2015.

REGISTRADOR DE TEMPERATURA E UMIDADE DATALOGGER | ITLOG80 |. Disponível em: http://www.instrutemp.com.br/instrutemp/produto/245/datalogger+de+temperatura+e+umidade+instrutemp+itlog80. Acesso em Junho 2015.

Sensor de Distância por ultrassom - HC-SR04 . Disponível em: http://www.labdegaragem.org/loja/sensor-de-distancia-por-ultrassom-hc-sr04.html. Acesso em Junho 2015.

Sensor de Movimento PIR. Disponível em: http://www.labdegaragem.org/loja/sensor-de-movimento-pir.html. Acesso em Junho 2015.

Socket.io. Disponível em: http://socket.io. Acesso em Junho 2015.

Sqlite. Disponível em: http://sqlite.org. Acesso em Junho 2015.

Story and History of Development of Arduino. Disponível em: http://www.circuitstoday.com/story-and-history-of-development-of-arduino. Acesso em Junho 2015

Termohigrômetro Data Logger USB até 6.000 Registros com Sensor Externo de Temperatura e Umidade Incoterm 3030.22 - Medidores no Pontofrio.com. Disponível em: http://www.pontofrio.com.br/Ferramentas/ferramentasdemedicao/Medidores/Termohigrometro-Data-Logger-USB-ate-60.000-Registros-com-Sensor-Externo-de-Temperatura-e-Umidade-Incoterm-3030.22-4087902.html. Acesso em Junho 2015.

Tutorial: Utilizando o GPS Shield como dispositivo de anti-evasão - Laboratorio de Garagem (arduino, eletrônica, robotica, hacking). Disponível em: http://labdegaragem.com/profiles/blogs/tutorial-utilizando-o-gps-shield-como-dispositivo-de-anti-evasao. Acesso em Junho 2015.